SKROLLI

2013.2
SKROLLI.FI

Tietokonekulttuurin erikoislehti

Ohjeet aikamatkaajalle:
Tietokoneen rakentaminen muinaisuudessa

Hackerspacet Suomessa:
Luovien nerojen pesimäpaikat

Palikasta botiksi:
Älyleegon tarina

SQL-injektiot

Dwarf Fortress
— Sano ystävä ja astu sisään

Hyi! Demoja!

Tie 3D-velhoksi alkaa:
OpenGL:n alkeet

Sisällysluettelo

Ville-Matias Heikkilä
päätoimittaja

Skrolli
Tietokonekulttuurin erikoislehti

Yhteydenotot toimitus@skrolli.fi
ircnet — #skrolli

Päätoimittaja Ville-Matias Heikkilä

Toimitus Lauri Alanko, Tapio Berschewsky, Jari Jaanto, Mikko Heinonen, Ronja Koistinen, Anssi Kolehmainen, Sade Kondelin, Ninnu Koskenalho, Toni Kuokkanen, Juho Lehtinen, Elias Linjama, Risto Mäki-Petäys, Oona Räisänen

Avustajat Markku Alanen, Eetu Korhonen, Matti Hämäläinen, Ville Lahdenvuo, Olavi Lintumäki, Juuso Metsänvuori, Tomi Pieviläinen, Visa-Valtteri Pimiä, Ville Ranki, Mikko Rasa, Jari Sihvola, Hannu Viitala

Kannen kuvat Risto Mäki-Petäys, Olli Oikarinen, Nuppu Oikarinen, Unna Oikarinen

Julkaisija Alternative Party ry

Painopaikka Tammerprint, Tampere,
ISSN 2323-8992 (painettu)
ISSN 2323-900X (verkkojulkaisu)

YMPÄRISTÖMERKKI
MILJÖMÄRKT
441 878
Painotuote

Skrolli kasvattaa selkärangan

Kuten näkyy, Skrolli ei jäänyt yhden numeron ihmeeksi. Ensimmäinen numero otettiin vastaan niin innostuneesti, että toiminnan ensimmäinen vuosi on taattu. Lehteä tekevä yhteisö on innostunutta ja sinnikästä porukkaa, mutta kuinka pitkälle pelkkä vapaaehtoishenki riittää?

Tietokonekulttuuri on täynnä yhteisöjä ja alakulttuureita. Jotkin ovat syntyneet yhden projektin ympärille, toiset taas laajemman ideologian. Monet yhteisöt ovat näivettyneet innokkaan alun jälkeen, muutamat ovat pysyneet elinvoimaisina vuosikymmentolkulla. Kuppikunnan kohtaloa on vaikea arvata ensivaiheittensa perusteella, ja niinpä Skrolli-projektiinkin on alusta saakka liittynyt pieni epävarmuus.

Skrolli lähti liikkelle yhteisövetoisena vapaaehtoisprojektina, johon ihmiset ovat tarjonneet sisältöä oma-aloitteisesti ja talkoohengessä. Projekti alkaa kuitenkin olla jo sen verran vakiintunut, ettei tulevaisuutta voi enää laskea pelkän luontaisen innostuksen ja hyväntekeväisyyden varaan.

On tullut aika vahvistaa tukirankaa. Lehden asioita hoitamaan perustetaan loppukesästä yhdistys, Skrolli ry. Lehdenteon prosesseja vakavoitetaan ja nimellisiä palkkioitakin ruvetaan maksamaan. Myös pitkään harkinnassa ollut keskustelufoorumi perustetaan irkkikanavan ohelle. Lehti ei kuitenkaan aio suistua kilpailijoidensa viitoittamalle tielle, vaan pysyy edelleen yhteisöllisenä. Tämä näkyy myös kakkosnumeron aiheissa.

Siinä missä ykkösnumeron jutut keskittyivät paljolti tekniikkaan ja historialliseen jatkuvuuteen, kakkonen painottaa luovaa yhteisöllisyyttä. Mitä hackerspaceissa tapahtuu? Millaisia taidemuotoja hakkerialakulttuureissa syntyy? Kuinka tekniikan rajat vaikuttavat yhteisön kehitykseen? Suomalaisen tietokonekulttuurin rönsyt johtavat hyvin usein demoskeneen, johon törmätään myös monissa tämän numeron jutuissa.

Monet saavat lehden käsiinsä sopivasti kesälomaksi, jolloin on aikaa askarrella yhden jos toisenkin tietokonekulttuurisen aiheen parissa. Tulkaa siis foorumille kertomaan tekemisistänne ja vahvistamaan samalla Skrolli-yhteisöä entisestään!

Desimaali	Binääri	Trinääri
-7	1001	-+-
-6	1010	-+0
-5	1011	-++
-4	1100	0--
-3	1101	0-0
-2	1110	0-+
-1	1111	00-
0	0000	000
+1	0001	00+
+2	0010	0+-
+3	0011	0+0
+4	0100	0++
+5	0101	+--
+6	0110	+-0
+7	0111	+-+

Skrollin ykkösnumeroa luettiin niin innokkaasti ja tarkasti, että saimme kuulla sivun 40 huolimattomuusvirheestä kymmeniä kertoja. Tässä siis trinäärilukutaulukko korjattuna. Hieno homma, jatkakaa samaan malliin!

Pelle Pelottoman työhuoneella
— hackerspacet ja itse tekemisen huuma

Hackerspacet ovat erilaisten projektien toteuttamiseen tarkoitettuja yhteistiloja, jotka ovat kaikkien jäsentensä käytettävissä. Niiden toiminnassa keskeistä ei välttämättä ole tekemisen muoto, vaan aito motivaatio ja kiinnostus tekemiseen. Hakkeria ajaa oppimisen halu ja sellaisten asioiden tekeminen, jollaisia ei kenties kukaan ole aiemmin tehnyt.

Teksti: Ville Ranki, Kuvat: Ville Ranki, Juuso Metsävuori

Tarve rakenteluun varatulle fyysiselle tilalle syntyy, koska erityisesti kaupungeissa on hankalaa löytää kotoa tarpeeksi työtilaa. Koneet ja laitteet maksavat, ja harrastelija tarvitsee monia niistä vain satunnaisesti. Lisäksi on mukava tavata samanhenkistä seuraa ja tehdä projekteja yhdessä. Suomessa on hackerspaceja tällä hetkellä ainakin seitsemässä kaupungissa.

Hackerspace on terminä hankala. Sana tulee verbinä tai substantiivina käytettävästä englannin kielen sanasta ”hack”, joka tarkoittaa hyvin laajasti kaikenlaista rakentelua softan kirjoittamisesta niksipirkkavirityksiin. Termi on syntynyt Massachusetts Institute of Technologyssa ja ollut käytössä jo 1960-luvulla. Suomessa hakkeri-termi on vakiintunut kuvaamaan tietoverkkoihin murtautujaa. Tässä artikkelissa ja hackerspace-skenessä hakkeri-sanaa käytetään alkuperäisessä merkityksessään.

Mitä haluaisit rakentaa tänään?

Yleisimmät hackerspaceissa tehtävät projektit liittyvät tietotekniikkaan, elektroniikkaan ja robotiikkaan. Olisi kuitenkin virheellistä kutsua hackerspacea tietotekniikka-, elektroniikka- tai robotiikkakerhoksi. Hackerspace tarjoaa jäsenilleen tilat tehdä, mitä itse haluavat. Mikään ei estä vaikka ruuanlaittoa tai sukkien virkkaamista, jos motivaatio on kohdallaan. Useimmat hackerspacet pyrkivätkin laajentamaan toimintaansa esimerkiksi musiikin, kuvataiteen, biotekniikan ja perinteisen käsityön suuntaan. Esimerkkeinä tästä Helsinki Hacklabin ommeltavan elektroniikan kurssit ja erilaiset elektronisiin soittimiin liittyvät työpajat.

> *Hakkeri on henkilö, joka tekee paahtoleipää kahvinkeittimellä.*
> – Wau Holland, Chaos Computer Clubin perustajajäsen

Hakkerointiprojekteja on ties kuinka monta sorttia. Utilitaariseen hakkerointiin kuuluu rikkinäisten laitteiden korjaamista tai uusien rakentamista rahan säästämiseksi, kuten esimerkiksi elektroniikan korjaus tai putkivahvistimen rakentaminen rakennussarjasta. Kun omiin tarpeisiin sopivaa tuotetta ei ole olemassa, se tehdään itse, ohjelmistoprojekteista huonekalujen kustomointiin. Rakentelu voi liittyä myös toiseen harrastukseen, ja valokuvauksen harrastaja saattaa rakentaa gyrovakauttimen tai kiskon timelapse-kuvausta varten. Taiteellista puoltaan pääsee toteuttamaan vaikkapa syntetisaattorin rakentamisella tai demokoodauksella.

Keskeistä toiminnassa on into oppia uutta. Vaikka lämpömittarin saa marketista pikkurahalla, on Arduinolla itse tehty tuhat kertaa hienompi ja opettaa uusia taitoja. Rakentelu ja kokeileminen on jo itsessään hauskaa, kunhan osia, työkaluja ja tekemistilaa on yllin kyllin. Saisiko ämpäristä ja savukoneesta tehtyä savurengaskanuunan kuten YouTubessa?

Hackerspace vs. Hacklab

Suomessa suurin osa hackerspaceista käyttää nimessään termiä Hacklab. Se on synonyymi hackerspacelle, kuten myös makerspace ja fablab. Hacklab-termi on suomalaisittain helpompi lausua, mutta hackerspace taas on maailmanlaajuisesti käytetympi ”brändi”. Nimi ei yhdistystä pahenna, ja sen keksimisessä voi käyttää luovuuttaan vapaasti.

Erityisten tilojen tarve

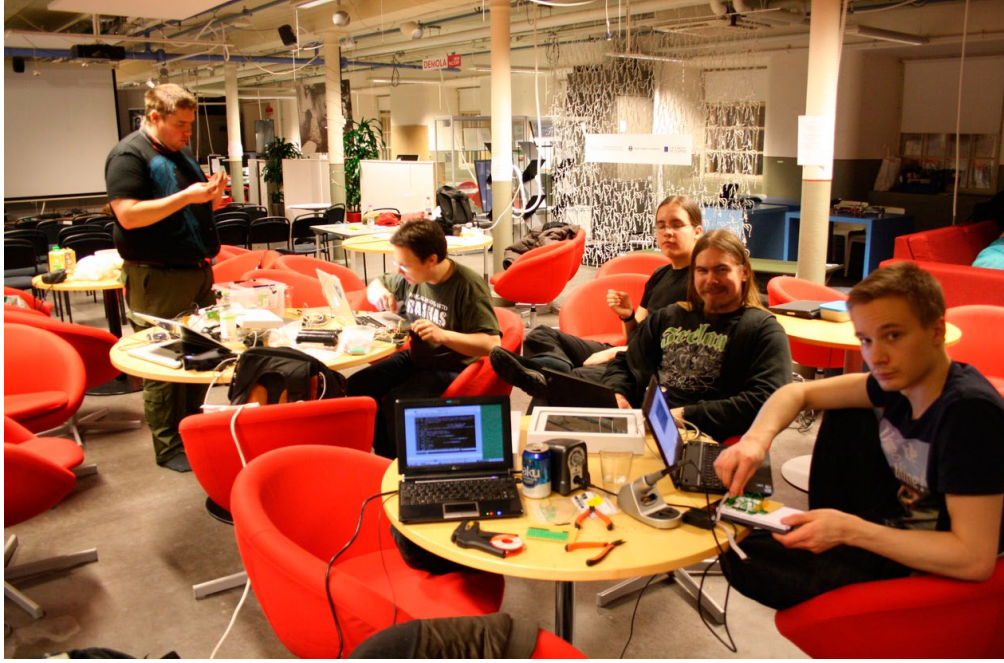
Joitain projekteja pystyy tietenkin tekemään kotonakin. Softan tekeminen ei vaadi muuta kuin tietokoneen, mutta kun

kuvioon lisätään elektroniikkaa tai mekaanista työtä, eivät tarvittavat työkalut enää mahdukaan olohuoneen pöydälle. Ahtaissa kaupunkiasunnoissa asuvilla ei yleensä ole mahdollisuutta perustaa kotiin säätönurkkausta, vaan projektit vaativat tavaroiden ja työkalujen esiin ottamista työn ajaksi ja varastointia sen jälkeen. Kynnys tekemisen aloittamiseen nousee, ja ensimmäinen askel jää usein ottamatta.

Mekaaninen työ aiheuttaa lähes aina myös ääntä. Kerrostalossa vannesahan käyttö tai rälläköinti eivät ainakaan paranna naapurisopua. Materiaalien sahaaminen, poraaminen ja laserointi aiheuttavat pölyä ja hajuja, mikä voi kiusata naapureita. Jopa harmittomalta vaikuttava 3D-tulostin pitää vanhaan matriisitulostimeen verrattavissa olevaa surinaa ja voi häiritä asuinkumppaneita.

Rajuimmillaan hackit sisältävät hitsausta, metallien valamista ja syövyttävien aineiden käsittelyä. Tällaiset toimet vaativat jo aivan omat tilansa ja välineensä. Esimerkiksi Lontoon hackerspacessa on oma varasto romupolkupyörille ja pyörän osille sekä hitsausvehkeet, joilla osista voi kasata uusia polkupyöriä.

Laaja komponentti- ja osavarasto on todettu käytännössä yllättävän hyödylliseksi. Tekemisen meininkiä tulee häkkäykseen aivan eri lailla, kun tarvittavat osat löytyvät saman tien hyllystä, eikä niitä tarvitse etsiskellä kaupasta tai tilata netistä. Suurinta osaa projekteista ei suunnitella etukäteen kovin tarkkaan, ja osahyllyn tarjonnasta löytyy uusia ideoita toteutukseen. Moni projekti saa alkunsa siitä, että osahyllystä löytyy jotain mielenkiintoista. Hackerspaceissa on usein läjä ”bonkkia” eli käytöstä poistettuja laitteita, joita ei-hakkeri kutsuisi elektroniikkajätteeksi. Hyvälaatuisesta bonkista saa irrotettua näyttöjä, kytkimiä, komponentteja, moottoreita ja muita käyttökelpoisia osia. Hackerspacen perustamisella saavutetaan luonnollisesti myös rahallista hyötyä: tulee halvemmaksi ostaa työkalut ja koneet yhteiskäyttöön sen sijaan, että jokainen hankkisi ne itselleen.



Hackerspace Summit Finland-tapahtumassa tavataan muiden kaupunkien hakkereita viikonlopun ajan Tampereella.

Kavereita ja koulutusta

Osalle hakkereita hackerspace on ennen kaikkea sosiaalinen tila, jossa tavataan kavereita ja tehdään yhdessä. Pajalla vietetyn ajan lisäksi samalla porukalla käydään syömässä, pidetään saunailtoja ja keksitään muuta ajanvietettä. Myös eri kaupunkien hakkereiden välillä on yhteistyötä, kuten Partyhat-projekti ja talvisin Tampereella järjestettävä Hackerspace Summit Finland -tapahtuma. Erilaisissa yleisötapahtumissa, kuten Wärk:Festeillä ja Model Expossa pidetään yhteistä hakkeriständiä. Hackerspacet ehkäisevät näin myös syrjäytymistä ja ”kybererakoitumista”. Hakkeritoiminnassa kenenkään ei tarvitse esittää vähemmän nörttiä kuin on.

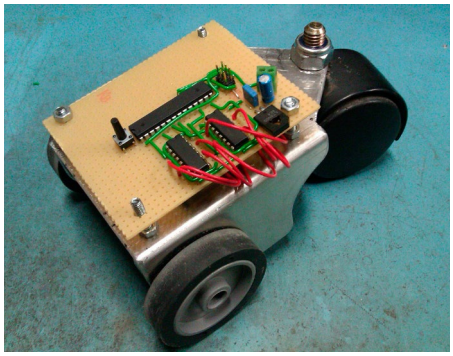
Koulutus on tärkeä osa hackerspacetoimintaa. Tyypillinen koulutusmuoto on omissa tiloissa pidettävä, tiettyyn aiheeseen liittyvä teemailta. Myös muille yhteisöille voidaan käydä pitämässä koulutuksia tai pyytää niistä joku kertomaan kiinnostavasta aiheesta. Esimerkkejä tyypillisistä koulutusteemoista ovat elektroniikan ja ohjelmoinnin perusteet, Arduino-rakentelu, 3D-tulostus ja Blender 3D:n käyttö. Koulutusten pitäminen vaatii jonkin verran vaivannäköä, mutta ne ovat hyvin suosittuja. Koulutusten pitäjän ei tarvitse aina olla alan asiantuntija, vaan riittää, että tietää aiheesta jotain ja on valmis jakamaan osaamistaan. Epämuodollisimmat koulutukset ovat ”workshoppeja”, joissa työskennellään ryhmissä ja opetellaan omatoimisesti aihetta workshopin pitäjän avustaessa ongelmatilanteissa.

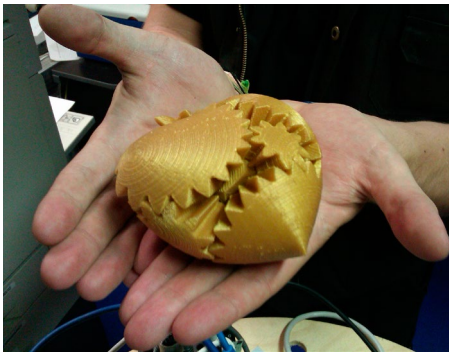
Hackerspacen ominaispiirteitä

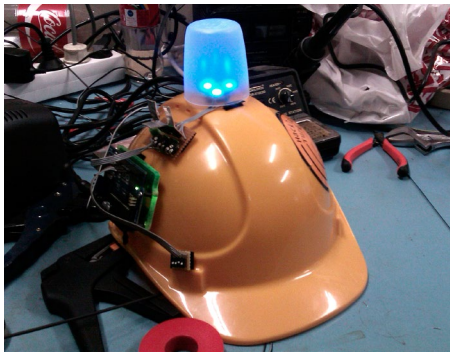
- **Avoimuus**
  Hackerspacen jäseneksi voi liittyä kuka tahansa. Esimerkiksi oppilaitoksissa ja yrityksissä on aina ollut labroja ja pajoja, mutta niiden käyttöoikeus on yleensä rajattu.
- **Autonomisuus**
  Hackerspace päättää omasta toiminnastaan ja on riippumaton ulkopuolisista tahoista, kuten vuokranantajasta tai sponsorista.
- **Tasavertaisuus**
  Hackerspaceissa vallitsee tyypillisesti melko matala hierarkia, ja jäsenet ovat keskenään yhdenvertaisia. Hallitus vastaa vain tärkeistä päätöksistä ja yhdistyksen rahankäytöstä. Isommista päätöksistä ja hankinnoista äänestetään. Jäsenet saavat omalla harkinnallaan ja vastuullaan esimerkiksi tehdä parannuksia tilaan keneltäkään kysymättä.

Kaupunki	Nimi	URL	Tilaa m²	Jäseniä	Jäsenmaksut*	Varustus
Helsinki	Hacklab Helsinki	http://hacklab.fi/	90	180	20€/vuosi, 40€/kk	Laserkaiverrin, 3D-tulostin, metallijyrsimen CNC-konversio työn alla
Tampere	Hackerspace 5w	http://5w.fi/	50	70	20€/vuosi, 30€/kk	3D-tulostin, laserkaiverrin, CNC-jyrsin
Turku	Hacklab Turku	http://hacklabturku.org/	50	20	20€/vuosi, 30€/kk	CNC-jyrsin, 3D-tulostin, pöytäsirkkeli
Jyväskylä	Hacklab Jyväskylä	http://hacklabjkl.org/	55	30	10€/kk/, 20€/kk	3D-tulostin (+ kaksi kesken)
Oulu	Tarlab	http://www.tarlab.fi/	30	15	20€/kk	Jäsenistöllä laserleikkuri ja muutamia 3D-tulostimia
Vaasa	Vaasa Hacklab	http://vaasa.hacklab.fi/	35	6	20€/vuosi, 30€/kk	3D-tulostin, puutyöverstas
Porvoo**	Porvoo H4cklab	http://hacklab.akan.fi/	?	?	0 €	?

* Vaihtoehdot ovat useimmiten vuosijäsenyys, jolla pääsee tiloihin silloin kun joku niissä on, ja kuukausimaksu, jolloin saa oman kulkuoikeuden. Työttömille, opiskelijoille ja eläkeläisille tarjotaan usein alennetut ”starving hacker” -hinnat.
** Tietoja ei saatu lehden painoon mennessä.







Näin perustat hackerspacen

1. Perusta ydinryhmä

Tarvitset 3–5 innostunutta ihmistä ja näkyvyyttä internetissä. Tee yksinkertaiset kotisivut ja postituslista. Hacklab.fi -domainin alle saa oman kaupunki.hacklab.fi -osoitteen. Lisää kaupunkisi hackerspaces.org:n listaan. Verkostoitukaa muiden hakkerien kanssa esimerkiksi irc-kanavalla #hacklab.fi, muissa hackerspaceissa ja talvisin järjestettävässä Hackerspace Summit Finland-tapahtumassa.

2. Kerää kiinnostuneita

Järjestäkää esimerkiksi kahvilatapaamisia, joissa tutustutte toisiinne. Mainostakaa paikoissa, joista voisi löytyä aiheesta kiinnostuneita. Oppilaitokset, harrastekerhot, työpaikat ja vaikka markettien ilmoitustaulut ovat hyviä paikkoja. Kerätkää yhteystiedot myös niiltä, jotka haluavat lähteä mukaan vasta sitten, kun tilat on hankittu.

3. Yhdistyksen perustaminen ja tukianomukset

Rekisteröityä yhdistystä tarvitaan tilin avaamiseen ja vuokrasopimuksen tekemiseen. Kopioikaa säännöt olemassa olevalta hackerspacelta, jotta ne hyväksytään kerralla. Viimeistään tässä vaiheessa on syytä aloittaa rahallisen tuen hankkiminen. Tässä auttavat suhteet, sinnikkyys ja hyvä tuuri. Ottakaa huomioon, että tukihakemuksen jättämisestä saattaa kulua hyvinkin pitkä aika siihen että rahat ovat tilillä. Toiminta on mahdollista käynnistää ilman ulkopuolista tukea, mutta tukien avulla pääsee paljon nopeammin vauhtiin.

4. Tilan hankinta

Kun kiinnostuneita on riittävästi, miettikää paljonko olette kuukausittain valmiita maksamaan jäsenyydestä. Tästä saatte käsityksen vuokranmaksukyvystänne. Kartoittakaa neliövuokrahintoja kohdealueella. Kaupungeissa tila kannattaa perustaa kävelymatkan päähän keskustasta, jotta kauempaa saapuvat pääsevät paikalle kohtuullisella vaivalla. Kyselkää kaupungilta, muilta harrastekerhoilta ja vastaavilta tahoilta vapaita tiloja. Ilmainen tai nimellisellä vuokralla tarjottava tila olisi tässä vaiheessa onnenpotku. Muistakaa sisällyttää budjettiin myös internet-yhteyden hankinta tilaan. Tilan tärkeimmät ominaisuudet ovat ympärivuorokautinen kulkumahdollisuus, lupa aiheuttaa melua ja wc. Jäsenmäärän kasvaessa sähköinen kulunvalvonta voi tulla kyseeseen.

5. Tilan varustelu

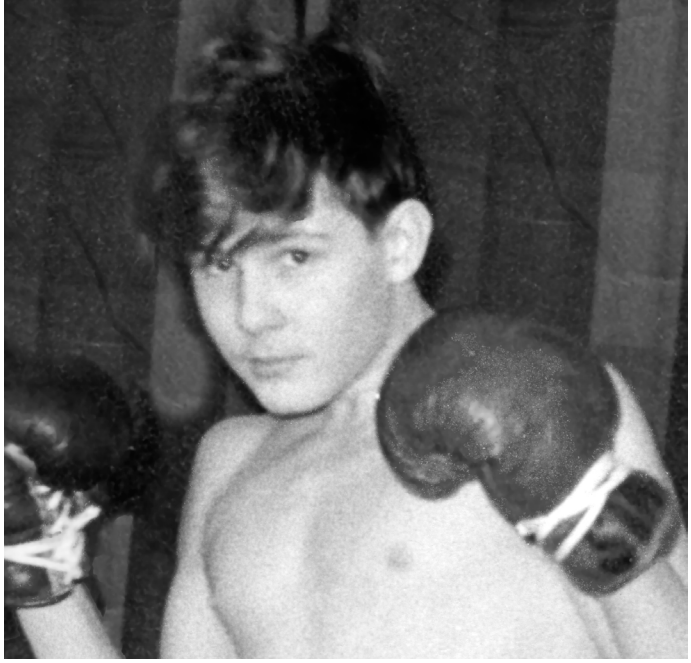
Alussa yhdistyksellä ei todennäköisesti ole varallisuutta, joten tilan varustus on jäsenten lahjoitusten varassa. Käytettyjä toimistokalusteita ja atk-laitteita löytyy ilmaiseksi, ja tila remontoidaan talkootyönä. Tässä vaiheessa jäsenet usein mielellään lahjoittavat tai pitkäaikaislainaavat omia työkalujaan ja materiaalejaan. Kalliimmat hankinnat (esim. 3D-tulostin) voidaan hankkia joko jäsenmaksuista saatavilla tuloilla tai hankinnasta kiinnostuneiden jäsenten kolehtina. Kolehtiin osallistumattomilta voidaan pyytää tarvittaessa pieni käyttömaksu.

6. Jäsenten hankinta

Alussa jäsenmäärän kasvattaminen on tärkeää. Varatkaa ständi mahdollisimman moneen tapahtumaan, jossa voisi olla potentiaalisia jäseniä. Erityisesti paikalliset tapahtumat ovat tärkeitä. Muistakaa myös ei-teknisille ihmisille suunnatut tapahtumat. Varautukaa siihen, että juuri kukaan ei ole kuullutkaan hackerspace-ilmiöstä, ja asia pitää kyetä selittämään lyhyesti ja helppotajuisesti. Kutsukaa paikallislehden toimittaja tekemään juttu toiminnastanne. Pitäkää viikottaiset tutustumispäivät (useimmilla hackerspaceilla tiistai-iltaisin) jolloin kuka tahansa voi tulla tutustumaan toimintaan. Järjestäkää avointen ovien päiviä, koulutuksia ja teemailtoja. Muistakaa esitellä toimintaa vieraillenne ja ottakaa uudet jäsenet alusta pitäen mukaan toimintaan.

Kotimaisia, isompia hakkeriprojekteja

- Partyhatwork — Yleisötapahtumissa pidettävä vapaamuotoinen hattu, joka viestii toisten hattujen kanssa XBee-verkon yli. Hatuissa on moniväriset ledit, ja niillä voidaan toteuttaa verkon yli synkronoituja ”demoefektejä”. Toteutetaan useamman hackerspacen yhteistyönä.
- Chernobyl-simulaattori — Helsinki Hacklabin astetta hullumpi neuvostoretroilu-multimediataideteos johon kuuluu mm. 2 m x 2 m ”reaktorikansi”.
- R100 — Sähköpyörätuolin runkoon tehty etäohjattu robotti, joka osaa kartoittaa ympäristönsä ja liikkua siinä itsenäisesti laserskannerin avulla.
- Jääkelkka — Leijapurjeen vetämä lumilaudasta ja neljästä suksesta koostuva jääkulkuneuvo.
- Imurirobotti-taideprojekti — Avustetaan Harri Larjostoa tilataideteoksessa, jossa modifioidut robottipölynimurit toimivat liikkuvana alustana.
- Metronäyttö — Tavoitteena on vähintään Tetris käytöstä poistetulla metronäytöllä. Tällä hetkellä ruudulle saadaan jo omia tekstejä ja tilanne kehittyy viikoittain.
- 24x16-ledimatriisi — 1,20 x 0,40 m. 16 kirkkautta/pikseli, 4 NES-ohjainta, i2c, käynyt Assemblyillä 2012 8x48-muodossa.
- Elovalo-valoteos, joka koostuu kolmesta 8×8×8 kuvapisteen LED-kuutiosta. Teos sijoitettiin Jyväskylän kirkkopuistoon osana Valon kaupunki -tapahtumaa syksyllä 2012. Kuutiot toimivat itsenäisesti akuilla ja kaikesta laskennasta vastaa AVR-mikrokontrolleri.
- LazorTouch — Tampereen Vapriikki-museolle tehty videoseinän ohjausohjelmisto, jonka avulla katsoja voi valita näytettävän videopätkän seisomalla vastaavan lattiaan maalatun lätkän päällä. Toteutettu laserskannerilla.



MikroBITTI in memoriam

Markku Alanen, X-päätoimittaja

Skrollin lukijoista useimmilla lienee jonkinlaisia omakohtaisia kokemuksia MikroBITTI-lehdestä. Aktiivisimmat olivat mukana Bitin tekemisessä jo vuoden 1984 ensimmäisestä numerosta alkaen — siis lukijat lehden teossa. Nykyisin tälläistä kutsutaan joukkoistamiseksi.

Bitille lukijoiden osallistuminen ei ollut ainoastaan lehden sisällön määrittämistä vaan myös sisällön tuottamista. Kasibittisiä ohjelmalistauksia kaseteilta purkaessaan toimitus ja avustajat olivat yötyöllistettyjä. Bitin mahtavan alkumenestyksen taustalla oli ensisijassa juuri uudenlainen lukijasuhde. Jako tyhmiin lukijoihin ja teräviin toimittajiin kuopattiin kuuden jalan syvyyteen. Oltiin kaikki enempivähempi samaan hiileen puhaltavaa porukkaa.

Bitti oli asemoitu harrastelehdeksi. Erikoista kuviossa oli se, että harrastuksesta tuli useille ammatti. Ajan mittaan yhä suurempi osa lukijoista oli joko valmiiksi tietotekniikkahommissa tai kovaa vauhtia matkalla moisiin. Moni lukija sai lehdestä enemmän potkua it-saloihin syventymiseen kuin konsanaan muusta atk-opastuksesta.

Vahva yhteisöllisyys piti samassa veneessä niin aloittelijat kuin edistyneemmät bitin nyplääjät. Bittiin mahtuivat niin kooderit, pelaajat kuin rakentelijatkin, vaikka välillä aiheuttivatkin ahdistusta toisilleen.

Taistelu lukijasta

Kova kilpailu lukijoista oli päivän sana tultaessa 1990-luvulle. Bitin asema oli tukala, ja lehden toiminnan lopettaminen oli hilkulla. Kun alkuvuosina vain Printti-lehti kalasteli menestyksekkäästi samoilla vesillä Bitin kanssa, niin nyt täytyi tilaajia jakaa niin Tiekkarille kuin mikroPC:llekin.

Osa Bitin lukijoista alkoi livetä ruodusta ja käydä vieraissa. Ammatillisten ambitioiden täyttyessä yhä useampi koki arvostuskysymyksenä "harrastelehden" nimeen vannomisen, kun tarjolla oli ammattilehtiäkin.

Ulkoisten kilpailijoiden lisäksi Bitti järjesteli kotoperäisesti markkinoille C=lehden Commodoren käyttäjille ja ilmeiselle kohderyhmälle Pelit-lehden. Kaikki nämä tavoittelivat osittain samoja lukijoita ja rokottivat tilaustuloja, joilla Bitin kulut kuitattiin. Tilanne oli se, että joko pillit pussiin tai levikki kasvuun. Vanhat konstit oli kaikki käytetty ja nyt oli keksittävä jotain tyystin uutta lukijoiden mielenkiinnon ylläpitämiseksi.

Verkkoa kohti

Internet oli harvojen ja valittujen temmellyskenttä 90-luvun alussa. Web oli vasta tuloillaan ensimmäisen Mosaic-selaimen myötä. Ei ollut irc-galleriaa tahi Facebookia. Pienet verkkoyhteisöt roikkuivat vielä puhelinlangoilla BBS-palvelimissa. Suunta oli kuitenkin selvä — Netti tulee ja Netti tappaa. Parempi siis hypätä rattaille mahdollisimman alkuvaiheessa.

Rakennettiin omin voimin MBnet — Bitin sähköinen ulottuvuus. Kaikki lukijat kutsuttiin mukaan maailman suurimpaan purkkiin, eräänlaiseen paikalliseen esi-internetiin. Nettiajan voidaan syystä katsoa alkaneen Suomessa 1994 hetkellä, jolloin puhelinverkot kaatuivat MBnetiin pyrkivien painosta.

MBnet ja Bitti muodostivat yhdessä uudenlaisen hybridimedian. Netti ja paperilehti pyrittiin nivomaan yhteen siten, että molemmat palvelisivat parhaalla tavalla lukijaa. Nettipuoli ei ollut mikään pakollinen kasvannainen vaan tasavertainen paperilehden kanssa.

Lehdessä käsiteltävien aihealueiden lukumäärä ei netin myötä suinkaan vähentynyt vaan kasvoi rajusti. Oli pakko lisätä sivumääriä ja suorittaa karsintaa vanhojen, vakiintuneidenkin sisältöjen suhteen. Lukijoiden hyväksyntä muutoksille ostettiin tarjoamalla heille riittävän paljon mielenkiintoista uutta sisältöä.

Maailman suurin

Uusi MikroBITTI osoittautui kaikilla mittareilla menestykseksi. Lukijat olivat tyytyväisiä ja saivat entistä helpommin äänensä kuuluviin netin välityksellä. Toimitus ja avustajat olivat ylpeitä aikaansaannoksistaan. Levikit kasvoivat hurjaa vauhtia ja kilpailijat karisivat kyydistä.

Bitistä kasvoi väkilukuun suhteutettuna maailman laajalevikkisin alan lehti. Toimittajat ja lehden avustajat edustivat alan terävintä kärkeä niin ammattitaidoltaan kuin lehden sisällön kehittämiseen sitoutumisessaan.

MikroBITTI oli yhteinen asia ja sen tekeminen oli hauskaa ja antoisaa. Palkitsevaa se oli myös lehden omistajalle. Lehden kate nousi 40 prosentin tuntumaan, jolloin esimerkiksi 10 miljoonalla pyöritettävä bisnes antaa voittoa 4 miljoonaa vuodessa.

Lehden talous ei ollut pätkääkään kiinni mainosmyynnistä — tilanne, josta muut lehdet saattoivat vain unelmoida. Sähköisellä puolella MBnet oli arvostetuimpien nettisaittien top kympissä.

Villi ja vapaa

MikroBITIN kustantajan ylivertainen tietotaito rajoittui lähinnä naisille tarkoitettujen lehtien julkaisuun. Bitti joutui jatkuvasti kamppailemaan oikeudestaan olla oma itsensä: erilainen, itsenäisesti ja luovan anarkistisesti toimitettu lehti.

MikroBITTI kehitti menestyksellisesti todella täysin uutta media-ajattelua. Perinteisen median päättäjille tämän uusmedian ymmärtäminen oli ylivoimaista. Pitkän väännön päätteeksi lehden journalistista linjaa alettiin harmonisoida niin sisällöltään kuin ulkoasultaankin kustantajan naistenlehtien kutyymiin. Viimeisen vinkaisun MikroBITTI päästi, kun se salvettiin ja päivänvaloon autettiin mB.

Muistakaamme kunnioituksella sitä MikroBITTIÄ, joka yritti olla lukijalle kunnon kaveri. Tänään Skrollilehti jatkaa noita kunniakkaita perinteitä. Skrollin sisältö kattaa vasta murto-osan mahdollisuuksistaan, mutta meno on sen verran ammattimaista, että lehdelle voi povata pitkää ikää. Uskon Skrollin lukijoiden omalla panoksellaan pitävän huolen vaivattoman vierinnän vakiintumisesta mikroharrastajien keskuudessa. Keep on Skrollin' 8-D

Saisinko rakennettua tietokoneen menneisyydessä?

Kuvitellaan, että joudut yhtäkkiä satoja vuosia ajassa taaksepäin. Saat jotenkin järjestettyä itsellesi suhteellisen mukavan elämän, mutta elämän tarkoitus puuttuu — nimittäin tietokone. Saisiko sellaisen rakennettua?

Teksti: Ville-Matias Heikkilä

Kuvat: Ville-Matias Heikkilä, Oona Räisänen, Wikimedia Commons (Klaus Nahr, Bruno Barral)

Nykyaikaiset tietokoneet ovat mikropiireihin perustuvia sähkölaitteita. On siis turhankin helppo olettaa, ettei tietokonetta pysty rakentamaan, ellei ensin ole kehitetty valtavaa määrää erilaisia välineitä ja tekniikoita eri elektroniikkakomponenttien valmistamiseen. Näin lohduttomasti asiat eivät sentään aikamatkaajalle ole. Ohjelmoitavan tietokoneen pystyy nimittäin aivan hyvin rakentamaan myös karkeammilla tekniikoilla, joita on ollut käytössä jopa vuosituhansien ajan.

Saksalaisinsinööri Konrad Zuse rakensi 1930-luvulla vanhempiensa olohuoneeseen ohjelmoitavan laskukoneen, joka tunnetaan nykyisin nimellä Z1. Se luki käskynsä reikänauhalta ja teki niitä noudattaen peruslaskutoimituksia 22-bittisille liukuluvuille. Lukuja pystyi tallentamaan työmuistiin, lukemaan käyttäjältä ja tulostamaan. Kaikki tietokoneen perusedellytykset siis löytyivät. Aikamatkaajan kannalta mielenkiintoinen piirre Z1:ssä on se, että se oli täysin mekaaninen: ainoa sähköinen osa oli sähkömoottori, joka piti koneistoa käynnissä.

Zuse rakensi Z1:n koneiston 30 000 ohuesta metalliliuskasta, jotka hän leikkasi kuviosahalla. Erimuotoisia liuskoja yhdistämällä hän pystyi rakentamaan loogisia portteja, joiden osat liikkuivat kahden mahdollisen asennon välillä muiden osien asennoista riippuen. Monimutkaisinkin binäärilogiikka perustuu pohjimmiltaan muutamaan erilaiseen loogiseen porttiin, joten liikkuvilla metallilevyillä pystyisi periaatteessa toteuttamaan mekaanisen vastineen mistä tahansa tietokoneesta.

Pikkutarkkaa mekaniikkaa osattiin rakentaa jo antiikin Kreikassa. Tähtitaivaan liikkeitä mallintanut Antikytheran kone perustui kolmeenkymmeneen pronssiseen rattaaseen, joista suurimmassa oli peräti 223 hammasta. Kreikkalaiset rakensivat myös automaattiteattereita, itseavautuvia ovia, kolikkoautomaatteja, höyrykoneita ja muita taianomaisia laitteita. Etenkin ensimmäisellä vuosisadalla elänyt Heron Aleksandrialainen kunnostautui hyödyttömien mutta hyvin kekseliäiden vempaimien luomisessa. Vaikuttaa siis selvältä, että kreikkalaiset olisivat pystyneet rakentamaan Z1:n, jos heille olisi annettu sen piirustukset. Koneen suunnittelu ei sen sijaan olisi onnistunut ilman symbolista algebraa ja binäärijärjestelmää, jotka saapuivat Eurooppaan vasta toisella vuosituhannella.

Rattailla vai vivuilla?

Metalliliuskoilla toteutettu binäärilogiikka on vain yksi monista tekniikoista, joiden varaan aikamatkaaja voi rakentaa tietokoneensa. Esimerkiksi pyöriviä rattaita ja askelrumpuja kannattaa myös harkita alkeisosiksi, sillä useimmat mekaaniset laskimet perustuvat niihin. Askelrummun (Staffelwalze) kehitti alkujaan Gottfried Leibniz 1600-luvulla omaa laskintaan varten.

Rattaita olisi käyttänyt myös englantilaismatemaatikko Charles Babbagen 1830-luvulla suunnittelema Analyyttinen kone, joka olisi valmistuessaan ollut maailman ensimmäinen ohjelmoitava tietokone. Se voisi periaatteessa toimia myös aikamatkaajan koneen mallina, kunhan sen rakennetta ei jäljittele turhan orjallisesti. Projekti kärsi nimittäin suuruuden-

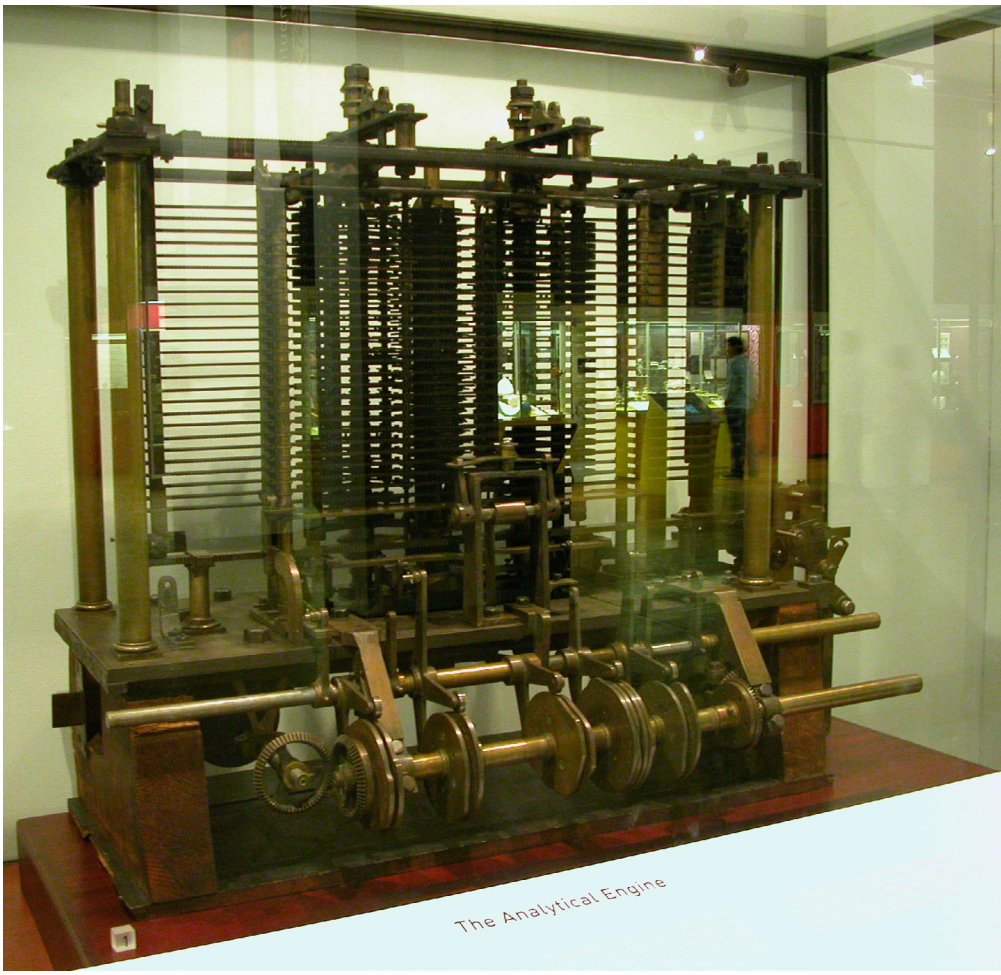
Uudelleenrakennettu Z1:n koneisto. Deutsches Technikmuseum, Berliini.

hulluudesta: esimerkiksi muistiin olisi pitänyt mahtua peräti tuhat 40-numeroista desimaalilukua, kun Zusellekin riitti vaivaiset 64 muistipaikkaa.

Vaikka Analyyttinen kone ei valmistunutkaan, Babbagen vaatimattomampi idea eli differenssikone muotoutui toimiviksi laitteiksi jo 1800-luvulla. Ruotsalainen Per Georg Scheutz rakensi differenssikoneestaan puisen prototyypin vuonna 1843 ja sai kaupaksi myöhemmin kaksi metallista versiota. Differenssikoneet eivät kuitenkaan ole ohjelmoitavia tietokoneita, vaan ne sopivat vain funktiotaulukoiden tuottamiseen. Toinen samalla vuosisadalla kehitetty tietokoneen esi-isä oli reikäkorttikone, jota käytettiin Yhdysvaltain vuoden 1890 väestönlaskennassa.

Tietokoneen fyysisenä perustana voi käyttää myös esimerkiksi köysiä, vieriviä kuulia tai pneumatiikkaa. Jos aikamatkaaja päättää keksiä sähkön, hän voi rakentaa releitä, elektroniputkia ja ehkä jopa puolijohdediodeja ja -transistoreita. Optisen tietokoneen toteuttamiseen tarvitsee sen sijaan melko varmasti laseria, joten sen voinee suosiolla unohtaa.

Ne, jotka haluavat kuivaharjoitella mekaanisen tietokoneen rakentamista ennen varsinaista aikamatkaa, voivat käyttää vaikkapa Lego-palikoita. Edellä



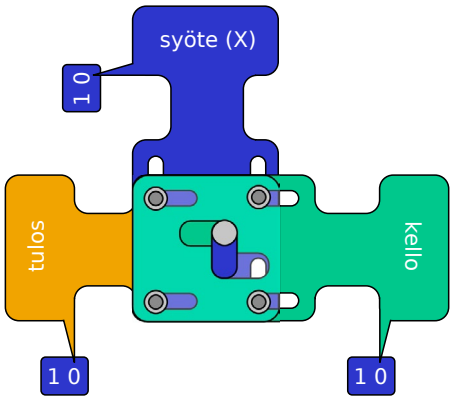
Pieni osa Analyyttisestä koneesta, Science Museum, Lontoo

mainitut Antikytheran kone ja Babbagen differenssikone on saatu toteutettua legoilla, ja brittiläinen legoharrastaja "Random Wraith" on rakentanut niistä myös loogisia portteja. Kokonaista ohjelmoitavaa tietokonetta ei sen sijaan kukaan liene vielä saanut aikaiseksi.

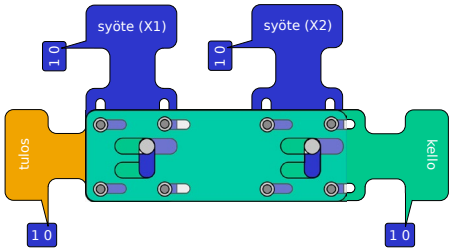
Yksinkertaisesta monimutkaiseen

Oli tietokoneen fyysinen ja teoreettinen perusta millainen hyvänsä, kannattaa se aina rakentaa abstraktiotaso kerrallaan. Ensimmäisessä vaiheessa kehitetään riittävä valikoima yksinkertaisia ja luotettavia alkeiselementtejä, esimerkiksi loogisia portteja. Toisessa vaiheessa näistä alkeiselementeistä rakennetaan hieman monimutkaisempia kokonaisuuksia, esimerkiksi muisti- ja yhteenlaskuelementtejä. Taso tasolta monimutkaisempiin yksiköihin edeten saadaan lopulta kokoon ohjelmaa suorittava kone, minkä jälkeen mahdolliset korkeammat abstraktiotasot voidaan toteuttaa ohjelmallisesti.

Yksinkertaisin looginen portti on NOT-portti, joka ottaa sisään yhden bitin (0 tai 1) ja antaa ulos vastakkaisen bitin (1 tai 0). Kuvassa 1 on esitetty Zusen Z1:n käyttämän NOT-portin toiminta. Porttiin sisään menevä bitti määräytyy sen mukaan, onko ylin levy ylä- vai alaasennossa. Vasemmanpuoleisen levyn asento osoittaa ulostulevan bitin. Oikeanpuoleinen levy on tarkoitettu tahdistukseen, jota ilman portti ei anna tulosta. Mekaaniset osat vaativat huolellista tahdistusta toimiakseen kunnolla yhdessä, joten tahtibittiä ei missään nimessä kannata optimoida pois.



Kuva 1: Z1:n NOT-portti



Kuva 2: Z1:n OR-portti

Toinen Zusen käyttämä portti oli OR (kuva 2), joka ottaa kaksi bittiä. Ulos tulee nolla, jos molemmat sisään tulevat bitit ovat nollia, ja muussa tapauksessa ykkönen. Binäärilogiikan toteuttamiseen riittää periaatteessa yksi ainoa porttityyppi (NAND tai NOR), mutta valikoiman kannattaa olla hieman laajempi, jotta koneen saa pienemmäksi. Esimerkiksi XOR-portti on hyvin käyttökelpoinen yhteenlaskupiireissä.

Jos halutaan laskea yhteen kaksi mielivaltaista binäärilukua, tarvitaan summaimeksi kutsuttu elementti (kuva 3), joka ottaa kolme bittiä (A, B ja C) ja antaa niiden summan kaksibittisenä (D ja E). Summan ylemmän bitin (D) voi ohjata toisen summaimen sisääntuloon, jolloin esimerkiksi kahdeksasta rinnakkaisesta summaimesta voidaan ketjuttaa kahden 8-bittisen luvun yhteenlaskupiiri (kuva 4).

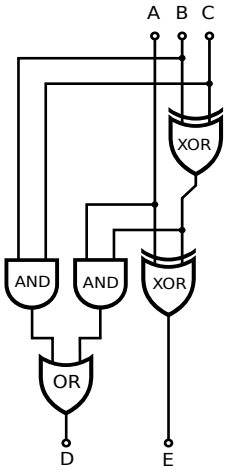
Pitkä summainketju ei kuitenkaan ole välttämätön. Esimerkiksi ensimmäinen suomalainen tietokone, ESKO (Elektroninen SarjaKOmputaattori), suoritti yhteenlaskut bitti kerrallaan yhtä summainta käyttäen. Myös PDP-8 -minitietokoneen pienin ja hitain malli toimii näin. Zuse ei kuitenkaan lähtenyt edes Z1:ssä tällaiseen nuukailuun.

Alkeet voi toteuttaa liikkuvien levyjen sijaan myös pyörivillä akseleilla ja rattailla, mikä saattaa olla jopa kannattavampi ratkaisu. Pyörimiseen perustuvassa logiikassa kannattaa Random Wraithin havaintojen mukaan käyttää alkeisoperaatioina pyörimismäärän summaa, erotusta, puolitusta ja itseisarvoa. Näistä analogisista operaatioista voi rakentaa melko yksinkertaisesti digitaalisia piirejä — niin loogisia portteja kuin binäärisummaimiakin.

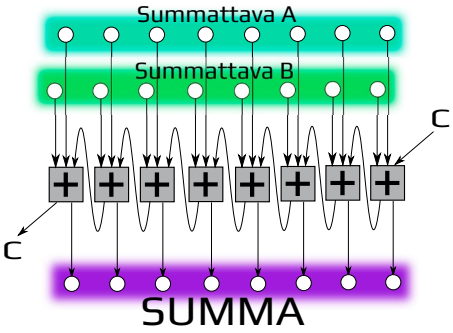
Muistit ja puskurit

Toimiva tietokone tarvitsee laskentaelementtien lisäksi tallennustilaa laskutuloksille. Z1:ssä oli tätä varten kaksi rekisteriä, R1 ja R2, ja 64 muistipaikan RAM-muisti. Laskutoimitukset tehtiin aina kahden rekisterin välillä, ja tulos tallennettiin jompaankumpaan. Muistin käsittelyyn oli omat käskynsä, jotka kopioivat dataa rekistereistä muistiin ja takaisin.

Z1:n RAM-muistissa kutakin bittiä vastasi pieni metalliliuska, jonka asentoa muutettiin. Liuskat oli asetettu ruuduksi, jota ympäröi valitsinmekaniikka. Kolme valitsimelle annettua bittiä valitsi tasolta yhden kahdeksasta rivistä ja toiset kolme yhden kahdeksasta sarakkeesta. Näin luku ja kirjoitus saatiin kohdistettua yhteen tason liuskaan kerrallaan. Muisti rakennettiin 22:sta tällaisesta bittitasosta, joita käytettiin samanaikaisesti.



Kuva 3: Summain, joka laskee yhteen kaksi binäärilukua.



Kuva 4: Summainketju, joka laskee yhteen kaksi 8-bittistä lukua.

Nykyaikaiset tietokoneet ajavat ohjelmansa RAM-muististaan, mutta Z1 luki käskynsä suoraan reikänauhalta, joka oli melko suosittu tallennusväline varhaisvuosikymmenten tietotekniikassa. Sekä reikänauhaa että reikäkorttia on kuitenkin käytetty erilaisten mekaanisten laitteiden toiminnan ohjaamiseen jo 1700-luvulta alkaen. Posetiiveissa ja soittorasioissa on käytetty myös mekaanista rumpumuistia, jolle voisi tietokonekäytössä tallentaa esimerkiksi mikro-ohjelman, joka toteuttaa koneen varsinaisen käskykannan.

Käskykanta

Z1:n käskykantaan kuuluivat aritmeettiset käskyt, muistinkäsittelykäskyt ja lukujen syöttö- ja tulostuskäskyt. Ongelmana oli kuitenkin hyppykäskyjen puute. Silmukat piti toteuttaa joko kirjoittamalla ne auki tai teippaamalla nauhan alku- ja loppupää yhteen. Oman koneen käskykantaa ei siis kannata suunnitella Z1:n kaltaiseksi, sillä kehittyneempiäkin esikuvia on tarjolla.

Yleiskäyttöisen tietokoneen saa periaatteessa jopa häkellyttävän yksinkertaiseksi. Esimerkiksi monet solu-

automaatit ovat Turing-täydellisiä. Tietokoneeksi riittäisi siis teoriassa laite, joka kelaa muistinauhaa ympäri loputtomiin ja muuttaa kunkin muistipaikan tilaa edellisten solujen tilojen pohjalta. Käytännössä tällainen kone olisi etenkin mekaanisena äärimmäisen hidas ja vaivalloinen, eikä siitä olisi paatuneimmallekaan idealistille kuin teoreettista iloa.

Monet alkeiskomponenteista asti tietokoneita rakentaneet harrastajat ovat päätyneet käyttämään PDP-8-minitietokoneen käskykantaa sen yksinkertaisuuden vuoksi, joten se voisi olla hyvä lähtökohta myös mekaaniselle tietokoneelle. Muita mahdollisia esikuvia voisivat olla Data General Nova, TMS1000, RCA 1802 ja MOS 6502. Korkeamman tason eleganssia kaipaavat voivat kehittää vaikkapa Forth-tyylisen pinopohjaisen käskykannan.

Keksisinkö sittenkin sähkön?

Mekaanisilla tietokoneilla on omat rajoitteensa. Vaikka sellaisen pystyisikin saamaan täysin luotettavaksi, on se moniin mielenkiintoisiin tehtäviin toivottoman hidas. Nopeimmat sähkömekaaniset tietokoneet ovat pystyneet noin kymmeneen yhteenlaskuun sekunnissa, joten tämän kummempaa suorituskykyä ei voi odottaa täysmekaanisiltakaan. Jo elektroniputkiin siirtymällä saadaan nopeus kuitenkin jopa monituhatkertaistettua.

Sähkön keksiminen voi olla aikamatkaajalle muutenkin kannattavaa. Sähkö mahdollistaa esimerkiksi lennättimen ja radion, jotka voivat antaa yhdelle jos toisellekin sotahullulle muinaishallitsijalle huomattavan etuaseman naapureihinsa nähden. Kun aikamatkaaja on viestintätekniikkaa myymällä päässyt hallitsijoiden suosioon, on hänellä hyvät edellytykset myös tietokoneen rakentamiseen.

Ensimmäinen sähkölaite, jonka aikamatkaaja ehkä haluaa tehdä, on paristo. Hänellä saattaa nimittäin hyvinkin olla mukanaan esimerkiksi matkapuhelin, joka on jo taustavalonsa vuoksi muinaisihmisille aivan uskomaton taikaesine. Sitä on siis hyvä päästä lataamaankin välillä, jotta taikavoiman eduista pääsee nauttimaan mahdollisimman pitkään. Elektrolyyttinä voi käyttää esimerkiksi hapankaalia, pihlajanmarjoja, sitrushedelmiä tai etikkaa. Lisäksi tarvitaan kahta erilaista metallia anodiksi ja katodiksi. Johdinten ja liittimen askarteluun kannattaa pyytää taitavan sepän apua.

Generaattorien, muuntajien ja releiden rakentamiseen riittävät käsityötaidot ovat olleet olemassa jo satoja vuosia, mutta fysiikan ymmärrys ei riittänyt laitteiden keksimiseen ennen 1800-lukua. Elektroniputken valmistaminen taas onnistunee lasin, metallin ja eristyksen osalta perinteisillä käsityötaidoilla, mutta riittävän tyhjiön saaminen putken sisään voi koitua ongelmaksi. Jonkinlaisia tyhjiöpumppuja tunnettiin jo antiikin aikaan, mutta putkitietokoneita havitteleva aikamatkaaja joutuu todennäköisesti kehittämään paremman tyhjiöntekotavan. Puolijohdekomponenteista kiinnostunut saattaa puolestaan joutua käyttämään jopa vuosikymmeniä tarvittavien prosessien kehittämiseen ennen kuin pääsee rakentamaan tietokonetta.

Kuinka oikeutan luomukseni?

Riittävän kyvykäs ja onnekas aikamatkaaja pystyisi siis todennäköisesti rakentamaan tietokoneen keskiajalla tai aiemminkin. Mutta kuinka muu maailma suhtautuisi moiseen vekottimeen?

Tekniikan ja taikuuden rajat olivat melkoisen häilyvät ennen valistusaikaa. Antiikin Kreikassa monimutkaisista koneista olivat kiinnostuneet lähinnä temppelit, jotka kilpailivat keskenään ja tarvitsivat siksi erilaisia "ihmeitä" vetonauloikseen. Italialaistiedemies Giambattista della Porta taasen vakuutteli 1500-luvulla teoksessaan Magiae Naturalis, että oikea kunnon magia perustuu luonnontieteisiin eikä sisällä mitään henkiolentoihin liittyvää. Aikamatkaajan on siis varauduttava siihen, että tiede, teknologia ja sarvipäisten pirujen kutsuminen ovat yhtä ja samaa noituutta valtaosalle ihmisistä.

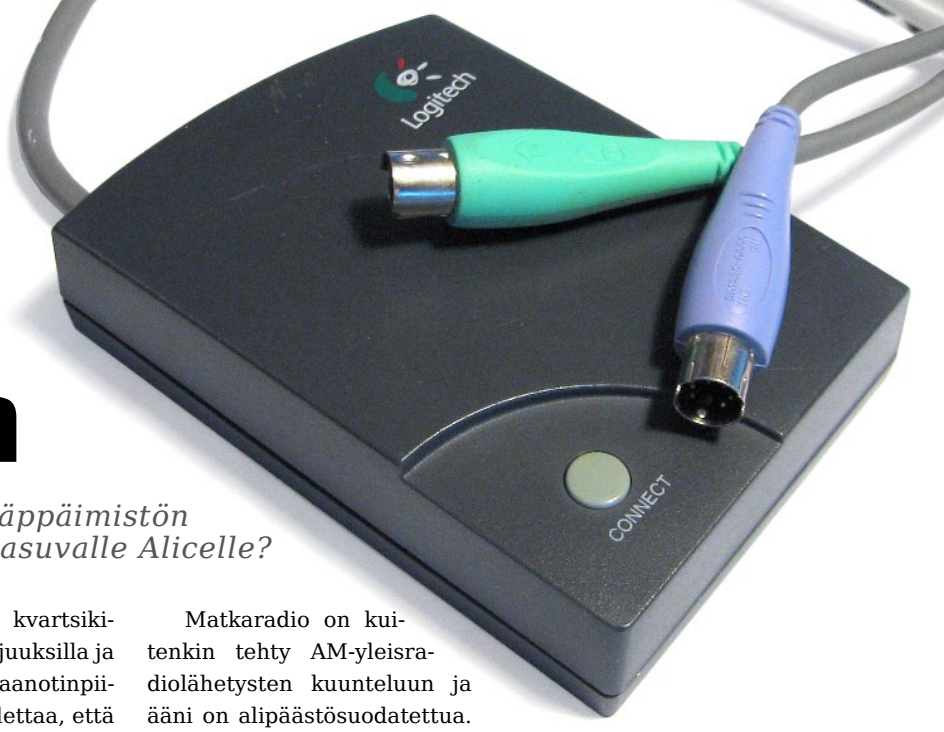
Tietotekniikan tärkeyttä ei myöskään kannata yrittää perustella järkiperäisesti. Suurin osa nykyihmisistäkään ei tajua yleisen tietojenkäsittelyn ideaa, sillä tietokoneet ovat heille pelkkiä sovelluskokoelmia. Idean vaikeutta kuvaa sekin, että alan pioneerien on aina ollut vaikea saada ideoilleen hyväksyntää. Leibnizin rakennettua toimivan laskukoneen kukaan ei jatkanut hänen työtään sataan vuoteen. Babbagen ideaa aivotyön automatisoinnista ei tajuttu, vaikka ruumiillista työtä automatisoitiin jo kovaa vauhtia. Edes 1970-luvulla tietokonefirmojen pomot eivät uskoneet, että kotitietokoneille olisi markkinoita.

Aikamatkaaja tuskin siis tapaa elämänsä aikana kovinkaan monia, joille pystyisi suurella vaivallakaan selittämään tietokoneen ideaa. Jotta tietokone ei jäisi pelkäksi henkilökohtaiseksi salaprojektiksi, on aikamatkaajan yritettävä myös muokata kulttuuria vastaanottavaisemmaksi sille. Yksi ratkaisu voisi olla filosofiksi ryhtyminen. Oppirakennelmiin voi ujuttaa viittauksia "ajatteleviin koneisiin" ja tietojenkäsittelyn teoriaan. Mikäli aikakauden ilmapiiri on turhan ahdasmielinen, ei radikaaleimpia ajatuksiaan kuitenkaan kannata kirjoittaa julki vaan tuoda ne esiin vain salaseurojen kokouksissa.

Mikäli aikamatkaaja joutuu menneisyyden sijaan tulevaisuuteen, jossa ihmiskunta on ottanut huiman teknologisen takapakin, on hänellä yksi oleellinen etu. Jos ihmisiä nimittäin on vielä olemassa, heillä on nykytekniikasta jäljellä ainakin jonkinlaista perimätietoa ja sitä kautta edellytyksiä sen hahmottamiseen. Uuskivikauden heimosoturi saattaa siis hyvinkin olla vastaanottavaisempi tietokoneen idealle kuin antiikin filosofi.

Tietotekniikan historiaa kerrotaan usein insinöörinäkökulmasta: mekaanisista osista on siirrytty tyhjiöputkien kautta transistoreihin ja niistä aina vain tiheämpiin mikropiireihin. Ajatusleikkimme osoittaa kuitenkin, että kulttuurin kehitys on ollut vähintäänkin yhtä tärkeää. Ihmiskunnan on pitänyt käydä läpi monta ajattelun murrosta, ennen kuin tietokoneen idea pääsi edes muodostumaan. Tietokone olisi siis ollut keskiajalla täysin anakronistinen käsittämättömyys riippumatta siitä, olisiko se rakennettu puusta vai kaukaisen tulevaisuuden komponenteista.

Ajatusleikin voi myös kääntää ympäri: jos keskuuteemme tulisi aikamatkaaja satojen vuosien päästä tulevaisuudesta, millaisina hän kokisi 2010-luvun tekniikan mahdollisuudet? Saisiko hän jo siitä aikaiseksi jotain meille täysin käsittämätöntä ja mullistavaa, vai katsoisiko hän viisaammaksi kehittää esimerkiksi nanoteknologiaa tai kvanttitietotekniikkaa eteenpäin ennen ihmekeksintönsä toteuttamista?



Salattuja purskeita

Kuinka turvallista on langattoman näppäimistön käyttäminen ilkeän Even naapurina asuvalle Alicelle?

Oona Räisänen

Moni tietokoneen oheislaite toimii nykyään langattomasti radioyhteydellä. Koska sivulliset voivat kuunnella kaikkea radioliikennettä varsin vaivatta ja huomaamattomasti, on liikenne yleensä tehty kuuntelukelvottomaksi salauksella. Mutta kuinka turvallista on vaikkapa langattoman näppäimistön käyttäminen ilkeän Even naapurina asuvalle Alicelle?

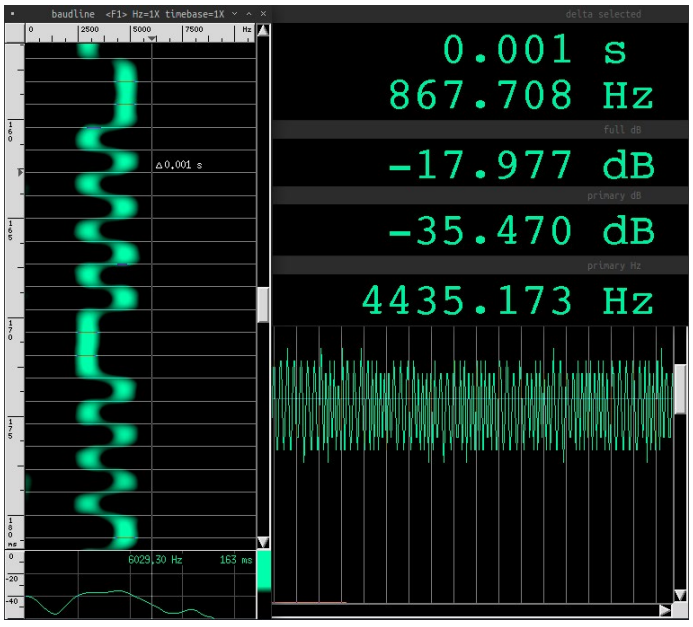
Etsin kokeilumielessä käsiini vanhan Logitech iTouch -näppäimistön vuodelta 2000. Se lähettää näppäinpainallukset 27 megahertsin taajuudella vastaanottimelle, joka liitetään PS/2-porttiin. Nykyaikaisiin USB-palikoihin verrattuna vastaanotin on suurikokoinen, ja sen toiminnasta ja taajuuksista voi päätellä jotakin jo piirilevyä tutkimalla. Piirilevyllä on mm. kvartsikiteitä eri ominaistaajuuksilla ja integroitu FM-vastaanotinpiiri. On siis aihetta olettaa, että tieto painetusta näppäimestä siirtyy radiokantoaallon taajuuden muutoksina.

Helpommin toiminta selviää kuuntelemalla radiota. 27 megahertsin taajuutta voi kuunnella millä tahansa matkaradiolla, jossa on 11 metrin lyhytaaltoalue. Signaali kuuluukin samassa huoneessa olevasta radiostani purskeina taajuudella 27,140 MHz ja on hyvin voimakas. Näppäintä painaessa kuuluu yksi purske, ylös nostaessa toinen. Lisäksi näppäimen pohjassa pitäminen tuottaa jatkuvan sarjan lyhyempiä ääniä. Näppäimistön sivussa on nappi, jossa lukee ”Connect”. Sen voisi kuvitella liittyvän salausavainten uusimiseen.

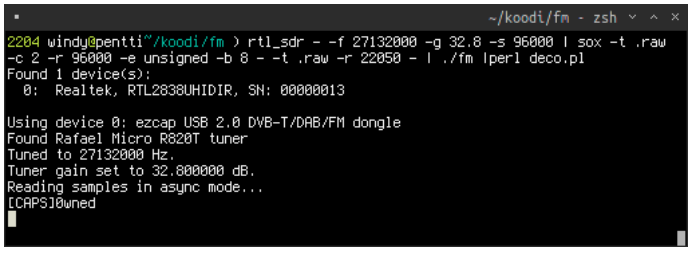
Matkaradio on kuitenkin tehty AM-yleisradiolähetysten kuunteluun ja ääni on alipäästösuodatettua. Osa informaatiosta saattaa olla kadonnut suodatuksessa. Paremman kaistanleveyden saan halvalla USB-digi-TV-sovittimella, jonka RTL2838-vastaanotinpiirin voi virittää yllättävän mielivaltaiselle taajuudelle RTL-SDR-ohjelmistoradion avulla. Käsken laitteen myös ohittamaan omat TV-purkupiirinsä ja kaatamaan raa’an näytevirran reaaliajassa tietokoneelleni. Näytevirta on radiosignaalin 8+8-bittinen vaihe-kvadratuuriesitys (I/Q), jonka reaaliosan voi tässä tapauksessa vähäisin vääristymin tulkita PCM:ksi ja tallentaa vaikkapa WAV-muotoon. Näin voin tarkastella ja käsitellä signaalia niin kuin se olisi ääntä.

Spektrogrammi näytevirrasta paljastaa, että signaali todella on FM-lähetteen erikoistapaus, binäärinen taajuudensiirtoavainnus (FSK) kahden kilohertsin siirtymällä. FSK:ssa kantoaallon taajuus vaihtelee kahden arvon välillä, joista toinen vastaa ykköstä ja toinen nollaa. Bittiviivoittimella mitaten näyttäisi, että datanopeus on noin 870 bps. Kirjoitan siis C:llä ohjelman, jossa on 870 hertsin sinioskillaattori ja vaihelukittu silmukka (PLL) pitämässä siniaaltoa lukittuna dataan. Oskillaattorin nousevan nollakohdan hetkellä ohjelma tulostaa nollan tai ykkösen riippuen signaalin senhetkisestä taajuudesta, joka saadaan Fourier-muunnoksella.

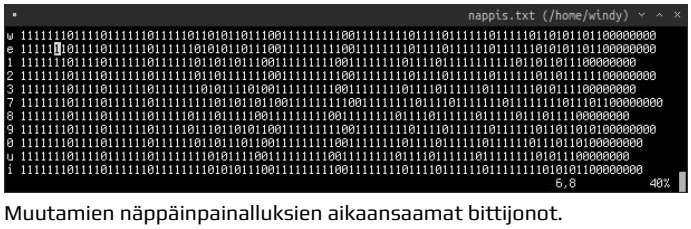
Yksi näppäimenpainallus tuottaa noin 85 bittiä dataa. Sanoman pituus vaihtelee hieman, joten on mahdollista, että FSK:n päällä on vielä jokin koodaus. Alkuosat biteistä näyttävät pysyvän aina sa-



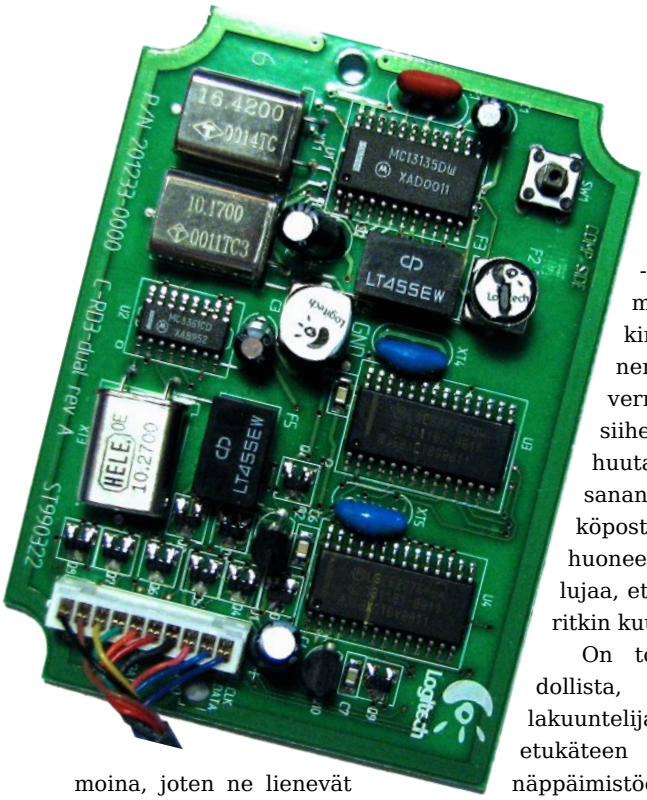
Raa'an näytevirran analysointia.



Ajossa valmis ohjelma, joka tulkitsee signaalin selväkieliseksi.



Muutamien näppäinpainalluksien aikaansaamat bittijonot.



Logitech iTouch -näppäimistöllä kirjoittaminen on siis verrattavissa siihen, että huutaisi salasanansa ja sähköpostiviestinsä huoneen yli niin lujaa, että naapuritkin kuulevat.

On toki mahdollista, ettei salakuuntelija pääse etukäteen käsiksi näppäimistöön eikä voi kokeilla kaikkia näppäimiä yksitellen. Tämä on kuitenkin pelkkä hidaste. Ilkeä Eve voi tilastoida vastaanottamiensa bittijonojen esiintymistiheydet ja verrata niitä vaikkapa suomen kielen tunnettuihin merkkitiheyksiin. Tämä tunnetaan frekvenssianalyysinä. Näin on mahdollista lopulta päätellä bittijonoja vastaavat näppäinpainallukset, vaikka protokollakin olisi tuntematon.

Nykyisin langattomat oheislaitteet ovat siirtyneet 27 megahertsistä 2,4 gigahertsin alueelle ja liikenteen salauskin on todennäköisesti kehittynyt. Taajuus ei toki vaikeuta kuuntelua eikä ole tietoturvaa parantava tekijä, joskaan juuri käyttämäni TV-vastaanotin ei ilman muutoksia aivan sinne asti virity. Jotkin oheislaitteet käyttävät salattua Bluetoothia tai muita yleisiä protokollia. Oma sohvanäppäimistöni käyttää Logitechin Unifying-tekniikkaa, joka väittää salaavansa liikenteen 128-bittisellä AES:llä. Sen avainneuvottelussa on kuitenkin mielenkiintoisen epäilyttäviä vaiheita, jotka saattavat tuoda tälle jutulle jatkoa.

moina, joten ne lienevät tahdistus-, osoite- tai tunnistekenttiä. Bittijonon loppuosa muuttuu näppäinpainallusten välillä, erityisesti jos painetaan eri näppäimiä. Se sisältää mahdollisesti PS/2-näppäinkoodin, mutta koodaus ei ole ilmiselvä.

Minkäänlaista salausta liikenteessä ei ole. Viestit nimittäin korreloivat voimakkaasti painetun näppäimen kanssa: tietyn näppäimen painamisen aiheuttaa aina suunnilleen samanlaisen bittijonon. Melko yksinkertainenkin salaus riittäisi piilottamaan korrelaation. Connect-nappi ei siis myöskään liity avainneuvotteluun.

Tuntemattomat muuttuvat kentät viesteissä ovat mahdollisesti laskureita ja tarkistussummia. Ne voidaan sivuuttaa, kun jokaisesta näppäimestä tallennetaan muutamia viestejä: viesteistä luodaan tyyppiesimerkki tai keskiarvo, ja nämä keskiarvot tallennetaan luetteloon yhdistettynä vastaavaan näppäimeen. Tämän jälkeen salakuunteleva Eve voi verrata kuulemaansa dataa kaikkiin luettelon bittijonoihin ja pisteyttää kaikki vaihtoehdot esimerkiksi Levenshtein-etäisyyttä käyttäen. Pienimmän etäisyyden saanut näppäin tulostetaan näytölle keylogger-tyylisesti.

Miksi demot ovat niin tympeitä?

Suomalaisessa tietokonekulttuurissa ei voi olla törmäämättä demoskeneen ja sen tuotoksiin. Monet eivät kuitenkaan oikein tajua demojen olemusta. Skrolli kertoo nyt, kuinka demoja kannattaa yrittää lähestyä, jotta niistä saisi jotain irti. Ville-Matias Heikkilä

Suomessa on väkilukuun nähden vahvempi demoskene kuin missään muussa maassa. Suurella osalla Suomen kovimmista IT-osaajista on demotaustaa, Suomen peliala on suorastaan demontekijöiden hallussa, ja merkittävimmät tietokoneharrastajien tapahtumat ovat pohjimmiltaan demoskenetapahtumia. Harvempi kuitenkaan osaa arvostaa demoja taidemuotona. Mitä kiinnostavaa on yksitoikkoisen musiikin tahtiin pyörivissä ja pomppivissa möhkäleissä? Eikö kukaan oikeasti kaipaa niihin mitään sanomaa tai juonta? Onko niissä oikeastikin jotain näkemisen arvoista salaseuroihin vihkiytymättömälle?

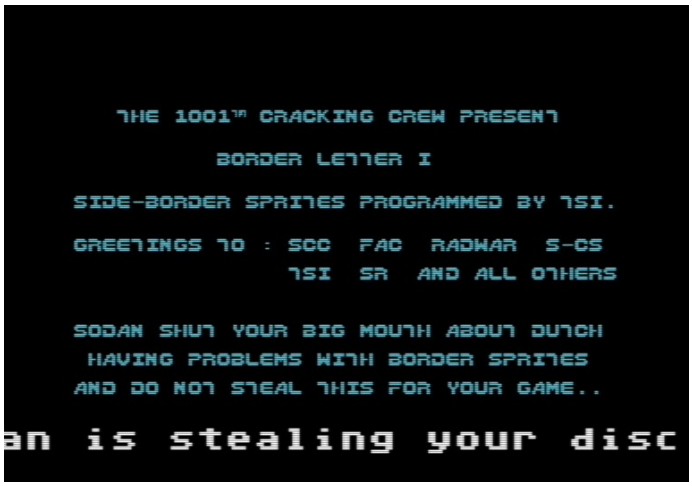
Osaamiskilpailu

Demot syntyivät nuorten tietokoneharrastajien halusta todistella osaamistaan toisilleen. 1980-luvun alun peleissä oli jo kopiosuojauksia, joten suojausten murtamistaito oli harrastepiireissä kovaa valuuttaa. Omat saavutukset piti tietenkin merkitä. Aluksi murtajat muokkailivat pelien teksteihin nimimerkkejään, mutta ennen pitkää myös signeerauksen näyttävyydestä tuli oma kilpailulajinsa. Näin kehittyivät crack-introt ja itsenäiset taidonnäyteohjelmat, demot.

Etenkin vanhemmissa demoissa suurimman osan elementeistä voi varsin turvallisesti lukea rehvasteluksi: "Katsokaa, osasin tehdä tämän!" Kovimmilla rajojenrikkojilla tämä voi myös kääntyä muotoon "Katsokaa, tämä on mahdollista!" Katsomiskokemukseen vaikuttaa paitsi käytettyjen teknisten puitteiden ymmärtäminen, myös tieto siitä, mitä kaikkea näiden puitteiden alaisuudessa on aiemmin tehty. Maallikko saattaa helposti hämmästellä täysin keskinkertaistakin neljän kilotavun demoa, sillä hänellä ei ole tuntumaa siihen, mitä tähän kokoon on aiemmin saatu mahtumaan.

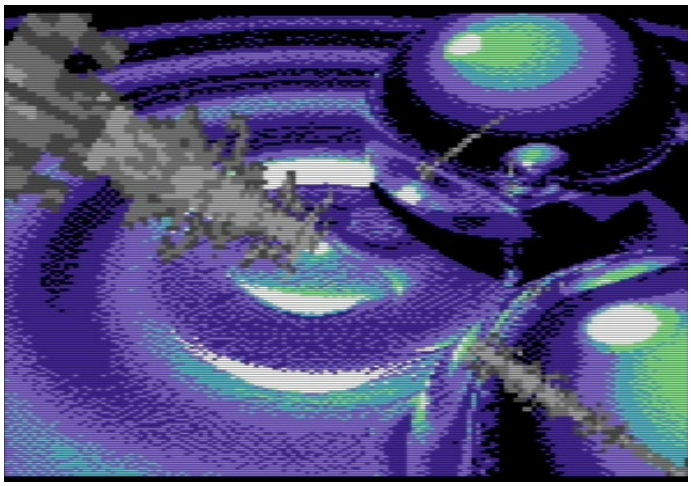
Osaamisella ja saavutuksilla kilpaileminen on edelleen tärkeä motivaattori demojen tekemiselle. Vaikka testosteroninkatkuisten jengisotien ajoista onkin jo aikuistuttu, valtaosa demoista julkaistaan edelleen demotapahtumien kilpailuissa eli kompoissa, joiden voittajat valitaan yleisöäänestyksellä. Yleisöön voi vedota monin tavoin, mutta teknisten rajojen rikkominen näyttävästi ja tyylillä toimii edelleen hyvin.

Nykykoneiden rajoja on kuitenkin vaikea saavuttaa, ja rimaa on lisäksi nostamassa peliteollisuus miljoonabudjetteineen. Demoskenen tekninen kekseliäisyys onkin vuosien mittaan siirtynyt megatavuluokan demoista tiukempiin kategorioihin, joissa on tyypillisesti rajoitettu ohjelman kokoa, laitealustaa tai molempia. Muutaman kilotavun kokoiseen ohjelmaan ei voi tallentaa valtavaa 3D-maailmaa tavanomaisin kei-

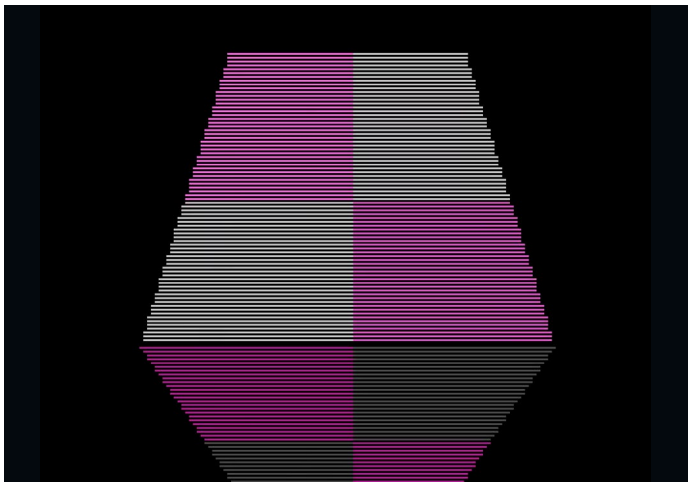


C-64-demo 1986: Sivureunat ylittävä skrolleri? En uskoisi, ellen itse näkisi! (1001 Crew: Border Letter I)

C-64-demo 2008: Sivureunat ylittävä plasmaefekti? Hohhoijaa, kyllä siinä on taas haaskattu aikaa rasteriajoitusten nysväykseen! (Booze Design: Edge of Disgrace)



PC-skenen saa naurunalaiseksi vaikkapa uudelleentoteuttamalla heidän yliarvostetuimman demonsa C-64:llä.
(Smash Designs: Second Reality 64, 1997)



Kun haastavimmatkin vakioalustat on jo koluttu, on aika nostaa vaikeustasoa.
(Trilobit: Doctor, Atari 2600, 2008)

noin, vaan sen luomiseen on kehitettävä ohjelmallisia tekniikoita. Kasibittisillä koneilla puolestaan ei voi tehdä kaikkea yleiskäyttöisellä 3D-moottorilla, vaan ennennäkemättömien asioiden tekeminen vaatii laitteiston ominaispiirteiden hyödyntämistä epätavallisin ja luovin tavoin. Joku onkin joskus verrannut demoskeneä ninjoihin: pakko mennä ikkunasta, vaikka vieressä on ovi.

Rajojen ja haasteiden tärkeys on luonut demoskenelle omanlaisen suhteen laitealustoihin. Tavanomaisessa tietokonekulttuurissa laitteiston ominaisuudet pelkistyvät aina lopulta laskentakapasiteetiksi, ja vanhat laitteet syrjäytyvät uusien ja tehokkaampien tieltä. Demontekijöille sen sijaan kaikki alustat ovat olemassa tässä ja nyt, kukin omine erityispiirteineen, rajoituksineen ja niistä kumpuavine haasteineen ja estetiikkoineen. Alustavalinta on myös pikemminkin makuasia kuin sukupolvikysymys: esimerkiksi 1990-luvulla syntynyt harrastaja saattaa hyvinkin käyttää demotaiteeseen mieluummin Commodore 64:ää kuin nyky-PC:tä, eikä siinä ole kenenkään mielestä mitään outoa.

Kauneuskin on tärkeää

Tekniset saavutukset ovat kuitenkin lyhytikäisiä. Joku vie tekniikkaa aina eteenpäin, eivätkä vanhat saavutukset enää nosta kylmiä väreitä pintaan. Vuodesta ja vuosikymmenestä toiseen katseltavat demot loistavatkin myös esteettisessä mielessä: hyvää musiikkia, nättiä grafiikkaa, hienosti musiikin mukana soljuvaa etenemistä. Useimmat nykydemot, varsinkin väljärajaiset PC-demot, eivät edes yritä päteä tekniikalla vaan lähtevät pelkästä estetiikasta. Demoista voi siis aivan hyvin nauttia myös tajuamatta niiden tekniikasta mitään.

Esteettinen ulottuvuus on ollut demoissa alusta asti. Monet pitkät skrollitekstit olisivat jääneet 80-luvulla lukematta, ellei taustalla olisi soinut hyvä musiikki eivätkä rasteripalkit olisi pomppineet kauniisti. Samaa osaa saattoi pitää pyörimässä vaikka tuntikausia, sillä katsoja siirtyi itse seuraavaan osaan esimerkiksi välilyöntiä painamalla. 90-luvun alkupuolella vallitsevaksi rakenteeksi vaihtui trackmo-tyyli, jossa musiikki, visuaalinen esitys ja uuden sisällön lataus levyltä tahdistettiin muutaman minuutin mittaiseksi musiikkivideomaiseksi kokonaisuudeksi. Valtaosa nykydemoistakin noudattaa trackmo-rakennetta.

Tavanomainen demoestetiikka ei avaudu useimmille kovinkaan helposti. Monet maallikot esimerkiksi yrittävät suhtautua demoihin lyhytelokuvina, jolloin ne vaikuttavat hajanaisilta ja sisällöllisesti tyhjiltä. Ne, jotka vastaanottavat demot musiikkivideoiden tai VJ-settien tapaan, saavat estetiikasta hieman enemmän irti. Demojen katsominen vaatii kuitenkin joka tapauksessa opettelua. Se on kuin totuttelua musiikkigenreen, jossa on kuuluvien soitinten lisäksi visuaalisia soittimia.

Genrestä voidaan puhua myös siksi, että useimmat demot ovat hyvin muotokeskeisiä. Samat vakiintuneet elementit ja tyyliseikat toistuvat teoksesta toiseen: muille ryhmille lähetetään terveisiä, kuutioita pyöritellään kunnianosoituksena vanhoille perinteille, ja kaikkea mahdollista tahdistetaan bassorumpuun. Monilla ryhmillä toki on pitkälle viety omintakeinen tyyli, mutta genrepiirteisiin ei siltikään voi olla törmäämättä.

Muotokeskeisyyden yhtenä alkusyynä voisi pitää kilpailuja, joissa teokset asetetaan paremmuusjärjestyk-

Minikokoisilla vuoristolentelyillä säväytetään vuosikymmenestä toiseen. Nykyään koko maisema pitää tuottaa shader-ohjelmointina.
(RGBA: Elevated, PC-4K, 2009)

Monet ovat pitäneet tätä venäläisdemoa humoristisena, mutta tekijä on vakavissaan poliittisen sanomansa kanssa.
(Cyberpunks Unity: R, ZX Spectrum, 2004).

Alkaako tämä edes koskaan? En tajua, diskataan! (Halcyon: Chimera, PC, 2002)

seen. Tyyliltään kokeellisempi demo ei välttämättä menesty, ja liiallista sisältökikkailua saatetaan pitää jopa halpana keinona voittaa teknisesti ansiokkaampi teos. Toisaalta etenkin monia vanhoja tekijöitä vaivaa myös kunnianhimon puute — heille on pääasia, että he ylipäätään saavat tehtyä jotain osoittaakseen olevansa edelleen hengissä.

Tarvitaanko sisältöä?

Monissa demoissa näyttää olevan tekniikan ja estetiikan lisäksi myös sisältöä: teemoja, juonia ja jopa sanomaa. Usein tämä on kuitenkin pelkkää silmänlumetta. Jos yleisössä on paljon demoihin vihkiytymättömiä, kannattaa demoestetiikkaa viedä hieman elokuvamaisempaan suuntaan, jolloin teos näyttää kertovan jostain. Tästä kerronnasta on kuitenkin yleensä suunnilleen yhtä turhaa etsiä merkityksiä kuin italodiskon sanoituksista.

Siitä, ovatko demot taidetta, kiisteltiin jo 90-luvulla. Monille demot ovat pikemminkin käsityötä, teknistä näpertelyä, jossa estetiikka syntyy tekniikan ehdoilla. Toisaalta on myös demoja, joissa taiteellinen ilmaisu on etusijalla ja tekniikkaa käytetään vain toteutusvälineenä. Valtaosa demoista sijoittuu jonnekin näiden ääripäiden väliin, joskin yleensä lähemmäksi käsityötä kuin taidetta.

Demotekniikka tarjoaa periaatteessa erinomaiset edellytykset myös sisältölähtöisille teoksille. Idean toteuttaminen nimenomaan demona ei ole kuitenkaan yleensä kovin järkevää, ellei taiteilija erityisesti halua sanoa tekniikkavalinnallaan jotain. Jos joku haluaa tehdä esimerkiksi tavallisia animaatioelokuvia, hänen kannattaa opetella mieluummin siihen tarkoitetun valmisohjelman käyttö kuin demokoodaus. Toisaalta jos idea on niin erikoinen, ettei mikään valmisohjelma taivu siihen kovin hyvin, voi demotekniikka olla hyvinkin varteenotettava vaihtoehto.

Ilmaisun rajat

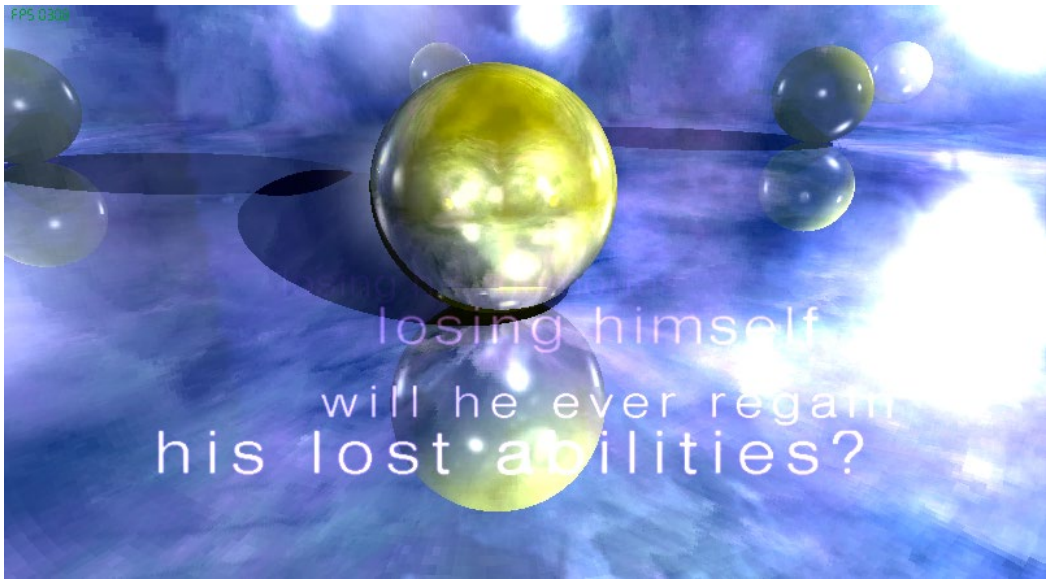
Teknisten rajojen rikkominen on yleisen näkemyksen mukaan koko demotaiteen järkähtämätön perusta. Tekniikkaa viedään aina uusiin äärimmäisyyksiin, saavutuksilla kilpaillaan ja uusien temppujen keksijät nostetaan kaikkia muita korkeammalle. Sisältö, muoto ja estetiikka tulevat ikään kuin sivutuotteina, eikä niiden rajoja rikota läheskään yhtä innokkaasti.

Ilmaisun rajojen hillitty puskeminen on kuitenkin aina kuulunut demotaiteeseen. Koko nykyinen demoestetiikka olisi jäänyt syntymättä, elleivät 90-luvun Amiga-ryhmät olisi rikkoneet ennakkokäsityksiä siitä, millaisia demot voivat olla. Sen sijaan jos demo poikkeaa hyvin voimakkaasti vakiintuneista normeista, voi vastaanotto olla hyvinkin nuiva. Monet demontekijät julkaisevatkin kokeellisemmat teoksensa eri taiteilijanimellä kuin "vakavammat" demonsa, jottei heidän maineensa tahriintuisi. Tyyliltään äärimmäisemmät demot ammentavat usein esimerkiksi noise- tai glitch-estetiikasta tai käyttävät esimerkiksi ambient-taustaa tavanomaisempien musiikkityylien sijaan. Estetiikkakokeiluissa on vain mielikuvitus rajana.

On kuitenkin yksi normi, johon avantgarde-henkisimmätkään demotaiteilijat eivät yleensä vaivaudu kajoamaan — nimittäin se, että demot ovat joka katsomiskerralla samanlaisia.

Pääsyy demojen itsepintaiselle staattisuudelle on varmastikin se, että ne tehdään kilpailujen kertaesityksiä varten. Mikään ei saa mennä esitystilanteessa vikaan, joten epävarmuustekijät on hyvä minimoida. Etenkin teknisille äärirajoille menevä koodi saattaa olla hyvinkin virhealtista, joten ohjelman suoritus on hyvä lyödä lukkoon. Lisäksi useimmat demokoodaajat ohjelmoivat myös työkseen, joten on ymmärrettävää, etteivät he halua keskittyä vapaa-aikanaan bugien metsästykseen ja poikkeustilanteiden huomioimiseen.

Satunnaisuutta ja vuorovaikutteisuutta olisi demoissa varmasti huomattavasti enemmän, mikäli niiden päänäyttämönä olisivat edelleen yksittäisten harrastajien ko-



Kaunis ja teknisesti vaikuttava PC:n 64K-demo raytracereineen, mutta pitikö se mennä pilaamaan jollain runontyngillä? Onneksi ne saa pois päältä. (Exceed: Heaven Seven, 2000)

tikoneet. Vähäisenkin dynaamisuuden kautta voisi myös hyvin tutkia niitä laajoja harmaita alueita, jotka jäävät demojen, pelien ja lelusovellusten väliin ja löytää niiden kautta aivan uudenlaisia rajoja rikottavaksi. Useimpien demontekijöiden päämotivaattorina ovat kuitenkin demotapahtumat vakiintuneine kilpailumuotoineen, eikä samantapaista intoa julkaista teoksia tapahtumien ulkopuolella ole.

Kun ulkopuoliset eivät ymmärrä

Demot voivat olla tympeitä monista syistä. Estetiikasta ei saa otetta, kaavamaisuus puuduttaa, ulkokuoren takaa ei löydy sisältöä, ja sisäpiirivitsit menevät yli hilseen. Mahdolliset tekniset saavutukset eivät avaudu tai vaikuttavat tyhjänpäiväiseltä näpertelyltä. Vaikka iso osa demoista onkin rehellisesti huonoja jopa tekijöidensä mielestä, maallikoiden on vaikea arvostaa hyvinäkään pidettyjä.

Alakulttuureille ominainen sisäänpäinlämpiävyys on ollut demoskenelle sekä siunaus että kirous. Toisaalta se on päästänyt demotaiteen kehittymään omilla ehdoillaan omaan suuntaansa, mutta toisaalta se on myös tehnyt siitä vaikeasti lähestyttävää ja muotoihinsa jämähtänyttä. Pahimmillaan skenen sisäiset normit estävät demoja etsiytymästä uusille urille ja demontekijöitä näkemästä laatikkonsa ulkopuolelle.

Kuppikuntaisuuden ongelmat on tiedostettu jo pitkään. Vuosituhannen vaihteessa, kun demoskenen kasvupiikki oli jo takana päin, alettiin demotaidetta tuoda ahkerammin julkisuuteen. Ruvettiin muodostamaan yhteyksiä taide- ja tiedepiireihin, kirjoittamaan kirjoja, järjestämään demoesityksiä ja seminaareja. Vuodesta 1998 järjestetty Alternative Party on alusta asti kyseenalaistanut demoskenen vallitsevia normeja ja yrittänyt löytää demotaiteeseen tuoreita näkökulmia.

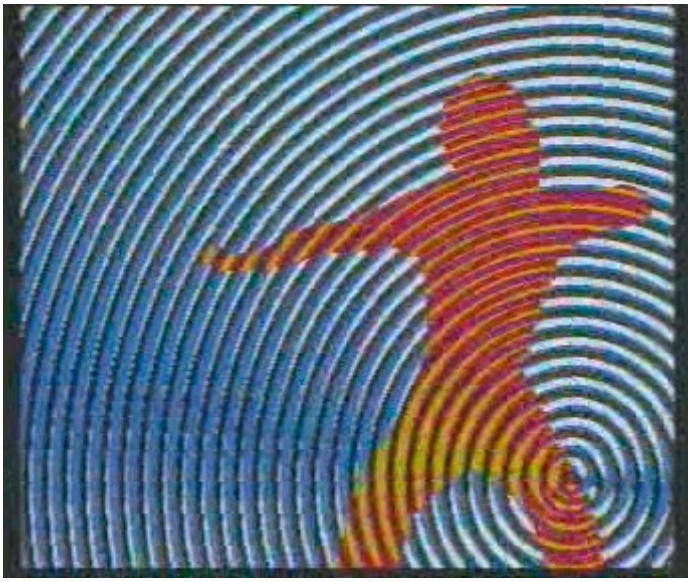
Vaikka ponnistelut ovatkin tuottaneet jonkin verran tulosta, perusongelma on pysynyt: demonteko on edelleen pitkälti sisäpiiritouhua, jota on vaikea sijoittaa toisten määrittelemiin lokeroihin. Demot ovat liian tyhjiä taiteeksi, liian outoja viihteeksi ja liian teknisiä akateemisiksi kiinnostuksenkohteiksi. Pelaajat eivät pääse pelaamaan niitä eivätkä hakkerit lukemaan niiden lähdekoodeja. Demontekijän voi olla yllättävän vaikea löytää yhteistä kieltä sellaisten ihmisten kanssa, jotka tekevät samantapaisia asioita muista lähtökohdista käsin. Jotta demotaide siis saisi ansaitsemansa hyväksynnän ja arvostuksen, tarvitaan tutkimusta, joka määrittelee sen suhteen muuhun maailmaan.

Demokulttuurin akateeminen tutkimus on alkanut toimia viime vuosina, kun tutkijat ovat löytäneet sen kuvaamiseen sopivia teoreettisia käsitteitä. Demojen estetiikasta tohtoriksi väitellyt Daniel Botz on esimerkiksi havainnut, etteivät laitealustat ole demontekijöille niinkään työvälineitä vaan raaka-ainetta, josta teokset muotoillaan. Digitaalisen median tutkijat Nick Montfort ja Ian Bogost ovat puolestaan perustaneet uuden tutkimusalan, alustatutkimuksen (platform studies), joka voi osoittautua hyvinkin hyödylliseksi myös demojen tutkimuksessa.

Demot ovat omaleimainen ja rajattomat mahdollisuudet tarjoava ilmaisumuoto, jonka kantavana voimana on radikaali tekninen kokeellisuus. Perinteet ja muotoseikat kuitenkin tukahduttavat demojen potentiaalia. Demojen tulevaisuuden kannalta olisi siis hienoa, jos niitä tehtäisiin, katsottaisiin ja ymmärrettäisiin enemmän myös perinteisen skenensä ulkopuolella. Tästä syystä myös Skrolli pyrkii pitämään demokulttuuria esillä.

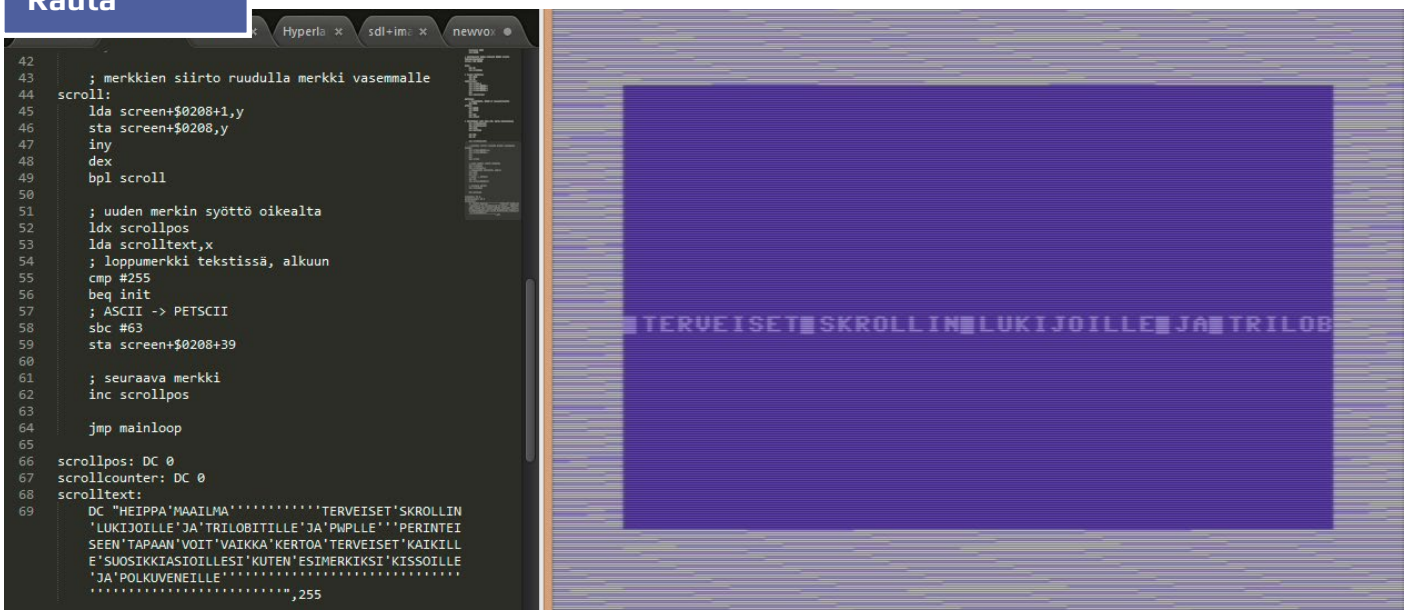
Terkut pitää toki lähettää perinteitä kunnioittaen, vaikka demo olisi rakennettu hc-punk-ääniraidan päälle. (Traktor: Jesus Christ Motocross, Amiga, 2009)

Tämä näyttää ensikatsomalta sanovan jotain, mutta oikeasti se on vain melko epäyhtenäinen sarja sekalaisia hienoja juttuja. (ASD: Lifeforce, PC, 2007)



Tätäkin klassikkoa paheksuttiin aikanaan, koska se keskittyy meininkiin enemmän kuin tekniikkaan. (Spaceballs: State of the Art, Amiga, 1992)

Teknisen osaamiskilpailun pyhää lehmää on myös välillä hyvä mäiskiä. (ISO: Vati, PC, 1997)



```
42
43      ; merkkien siirto ruudulla merkki vasemmalle
44  scroll:
45      lda screen+$0208+1,y
46      sta screen+$0208,y
47      iny
48      dex
49      bpl scroll
50
51      ; uuden merkin syöttö oikealta
52      ldx scrollpos
53      lda scrolltext,x
54      ; loppumerkki tekstissä, alkuun
55      cmp #255
56      beq init
57      ; ASCII -> PETSCII
58      sbc #63
59      sta screen+$0208+39
60
61      ; seuraava merkki
62      inc scrollpos
63
64      jmp mainloop
65
66  scrollpos: DC 0
67  scrollcounter: DC 0
68  scrolltext:
69      DC "HEIPPA'MAAILMA''''''''''''TERVEISET'SKROLLIN
        'LUKIJOILLE'JA'TRILOBITILLE'JA'PWPLLE'''PERINTEI
        SEEN'TAPAAN'VOIT'VAIKKA'KERTOA'TERVEISET'KAIKILL
        E'SUOSIKKIASIOILLESI'KUTEN'ESIMERKIKSI'KISSOILLE
        'JA'POLKUVENEILLE''''''''''''''''''''''''''''''''
        ''''''''''''''''''''''''''''''''",255
```


Homebrew

— kotipolttoista kehitystä konsoleille ja kotitietokoneille

Omien ohjelmien ja pelien luonti menneiden konsoli- ja tietokonesukupolvien koneille on varmasti käynyt monen laitteita käyttäneen mielessä. Julkaisuajankohtanaan suljetut tai muuten vain huonosti dokumentoidut koneet pysyivät saavuttamattomissa monelle innokkaalle ohjelmointiharrastajalle vuosia tai jopa vuosikymmeniä.

Teksti: Visa-Valtteri Pimiä

Kuvat: Visa-Valtteri Pimiä, Marko Mäkelä, Wikimedia Commons (Hedning, Appaloosa)

Kodin tietokoneet ja pelikonsolit yleistyivät 1980-luvun alkupuolella. Etenkin konsoleille on ollut perinteisesti liki mahdoton kehittää omia ohjelmia, sillä alustat ovat olleet suljettuja muille kuin kaupallisille peli- ja ohjelmistoyrityksille. Kotitietokoneiden ohjelmointi on ollut huomattavasti helpompaa, mutta siihenkin on liittynyt paljon salaisuuksia. Suljettujen laitteiden tapauksessa ihmisiä yhdistävä internet mahdollisti kuitenkin vapaammaan tiedonvaihdon. Yhteisöllisyyden lisääntyessä harrastelijaryhmien toiminnassa myös vanhat konsolit ja tietokoneet saivat ensimmäiset tietopankkinsa. Näistä ammennettiin pohja emulaattorien ja kääntäjien, manuaalien ja tutoriaalien, ohjelmaesimerkkien ja työkaluohjelmien kehitykselle. Nämä tarpeelliset osat loivat pohjan ns. homebrew-liikkeelle, jossa nimensä mukaisesti kotipolttoista ohjelmistoa luodaan omin ehdoin.

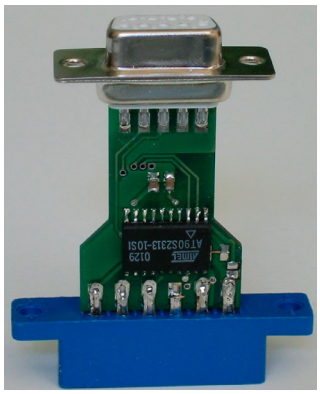
Nykyaikainen ohjelmistokehitys on yhä enemmän kytköksissä vapaasti saataviin kehitystyökaluihin ja yleisiin standardeihin, jotka ovat kaikkien saatavilla ja hyödynnettävissä. Tämä nykypäivän itsestäänselvyys ei kuitenkaan päde suljettujen alustojen, kuten koti- ja käsikonsoleiden, tai muuten vain nykylaitteiden kanssa suoraan epäyhteensopivien, noin vuosikymmenen takaisten ja sitä vanhempien alustojen kohdalla.

Erityisesti konsoleiden valmistajat ovat perinteisesti kontrolloineet sitä, millaiset kehittäjät ovat saaneet mahdollisuuden käyttää sisäiseen käyttöön suunnattuja kehitystyökaluja ja tarpeellisia julkaisukoneistoja ohjelmien saattamiseksi markkinoille asti. Mutta miten onnistuu ohjelmien kehittäminen laitteille, joille ei ole saatavilla virallisia kehitystyökaluja? Entä jos laite yksinkertaisesti on jo sen verran vanha, ettei sille ole olemassa mitään alkuperäisen kehittäjän resursseja? Olisiko myös mahdollista käyttää myös korkeamman tason ohjelmointikieliä ohjelmistokehitykseen näille erittäin yleisille kotitietokoneille ja samantyyppisille pelikonsoleille?

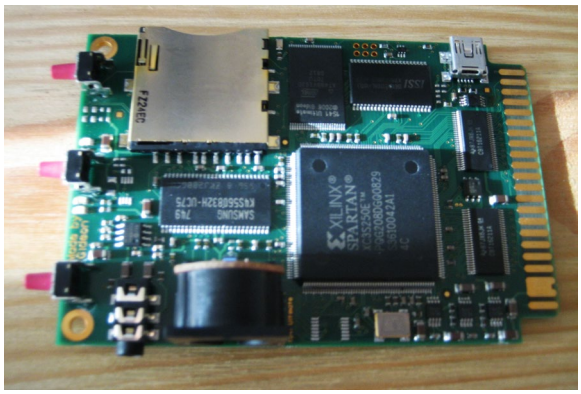
Tämän artikkelin tarkoitus ei ole olla syväluotaava "tee perässä" -opas. Pikemminkin se antaa tarvittavat ainekset lisätiedon hankintaan ja kattavan yleiskatsauksen homebrew-kehitykseen syventyen niihin osa-alueisiin, joissa vaaditaan erityisiä laitteisto- ja ohjelmistoratkaisuja.

PowerPak, CompactFlash-moduulisovitin Nintendo Entertainment Systemille.

IDE64, ATA-/ATAPI-sovitin Commodore 64:lle



C2N232, RS-232-liitäntäinen kasettiasemasimulaattori



1541 Ultimate Plus, FPGA-pohjainen levyasema- ja moduulisimulaattori Commodore 64:lle

Commodore 64 ja NES

Käsitellään ensimmäisinä esimerkkeinä kahta 8-bittiseen prosessoriarkkitehtuuriin perustuvaa suosittua kotikäyttöön tarkoitettua laitetta, Nintendo Entertainment Systemiä eli tutummin NESiä, ja Commodore 64 -kotitietokonetta. Kun lähtee selvittämään kehitystä tuntemattomalle alustalle, kannattaa aloittaa tunnistamalla sen oleellisimmat komponentit ja niiden ominaisuudet: prosessori, muistin määrä ja tyyppi, näytönohjain ja äänentoisto-ominaisuudet, tallennusmedia ja tiedonsiirto- ja syöttölaiteliitännät.

Listaamalla molemmista alustoista ominaisuudet voidaan niistä rajata tarpeellisten työkalujen etsintä oleellisiin valintoihin.

Kääntäjä

Monesti koodin voi kääntää kohdelaitteessa, mutta yleensä kannattaa käyttää ns. ristiinkääntäjää, jota ajetaan suorituskykyisemmässä tietokoneessa. Tämä nopeuttaa kääntämistä ja helpottaa ohjelmakoodin testaamista ja virheiden etsintää, ja modernien käyttöjärjestelmien tarjoamat edut ovat käytettävissä. Siksipä sen sijaan, että ohjelmakoodi käännettäisiin kohdealustalla, se käännetäänkin toisella koneella ja käsitellään sitten kohdealustan ymmärtämään suoritettavaan tiedostomuotoon.

Dasm on pitkän kehityksen tulos ja yhä varsin käyttökelpoinen assembler-ohjelma. C64:lle erikoistunut Kick Assembler sisältää monimutkaiset makro-ohjelmointiominaisuudet ja kattavan apukirjaston funktioita datatiedostojen kuten äänen ja grafiikan käsittelyyn. Muita suosittuja 6502-ristiinkääntäjiä ovat ACME, XA65 ja C-kääntäjä CC65:n mukana tuleva CA65. 6502-suoritinta ohjelmoidaan yleensä joko suoraan 6502-assemblyllä tai sopivalla C-kääntäjällä.

Commodore 64 lataa ja suorittaa binääritiedostoja sellaisenaan alkaen osoitteesta, joka annetaan tiedoston kahdessa ensimmäisessä tavussa. NES-emulaattorit käyttävät usean eri standardin mukaisia ROM-moduulien kuvaustiedostoja, jotka sisältävät ohjelmakoodin ja -datan lisäksi tietoja moduulin lisäominaisuuksista kuten lisämuistin määrästä ja lisäpiireistä.

Lisäpiireillä sekä Nintendo että kolmannen osapuolen kehittäjät laajensivat NESin ominaisuuksia sen menestyttyä markkinoilla. Commodorelle taas oli saatavilla hyvin paljon erilaisia laitteen toimintaa laajentavia moduuleja ja oheislaitteita, ja se oli avoimena julkaisualus-

	Commodore 64	NES
Suoritinarkkitehtuuri	6502-yhteensopiva	6502-yhteensopiva
Muisti	RAM: 64 kilotavua (+laajennuksia -BASICin syömä pieni osuus)	RAM: 2 kilotavua (+kasettilaajennukset)
Näytönohjain	VIC-II — useita erilaisia näyttötiloja erilaisilla värimäärillä, 8 pikselin rautaskrollaus, rautaspritet	RP2C02 — pehmeä nelisuuntainen skrollaus riippuen asetuksista, rautaspritet, usean väripaletin tuki
Äänentoisto	SID — erittäin monipuolinen syntetisaattoripiiri joka mahdollistaa monimutkaisen ääniohjelmoinnin	APU 2A03 — viisi kanavaa, 7-bittinen äänisampletoisto
Tallennusmedia	C-kasetti ja 5¼ tuuman levyke. Lisäksi yleisesti saatavilla olevia moduuleja, joissa tallennusominaisuudet.	ROM-kasetit ja Famicom Disk System-levykeasema, lisäksi saatavilla massamuistilaitteilta ROM-tiedostoja lukevia FPGA-pohjaisia flashcartteja.
Tiedonsiirto ja syöttölaitteet	Useita standardoituja tiedonsiirtoliitäntöjä, Atari-standardin peliohjainliitännät	NES-peliohjainliitännät tai Famicomin konsolissa kiinni olevat ohjaimet. Saatavilla lisäksi CopyNES-kehityslaite joka mahdollistaa nopean tiedonsiirron.

```
/* ESIMERKKIKOODIA COMMODORE 64:LLE -
   SKROLLIN TEKSTISKROLLERI
   KÄÄNTÄJÄ: DASM
*/
        processor 6502
        org    $1000

; näyttömuisti alkaa kohdasta $0400
; (osoite heksadesimaaleina)
screen: EQU $0400

init:
        ldy #0
        sty scrollpos

; ruudun tyhjennys
        lda #$20
        ldx #0
clearscreen:
        sta screen,x
        sta screen+$0100,x
        sta screen+$0200,x
        sta screen+$0300,x
        dex
        bne clearscreen

mainloop:
        ; reunaefekti, $d020 on reunaväriosoite
        ldx #$4d
effect:
        asl $d020
        inc $d020
        dex
        cpx #15
        bne effect

; skrollataan vain joka 120. kerta mainloopissa
        inc scrollcounter
        lda scrollcounter
        cmp #120
        bne mainloop

        ldx #38
        ldy #0

        sty scrollcounter

        ; merkkien siirto ruudulla yksi
        ; merkki vasemmalle
scroll:
        lda screen+$0208+1,y
        sta screen+$0208,y
        iny
        dex
        bpl scroll

        ; uuden merkin syöttö oikealta
        ldx scrollpos
        lda scrolltext,x
        ; loppumerkki tekstissä, alkuun
        cmp #255
        beq init
        ; ASCII -> PETSCII
        sbc    #63
        sta screen+$0208+39

        ; seuraava merkki
        inc scrollpos

        jmp mainloop

scrollpos: DC 0
scrollcounter: DC 0
scrolltext:
        DC "HEIPPA'MAAILMA''''''''''TERVEISET
'SKROLLIN'LUKIJOILLE'JA'TRILOBITILLE'JA'PWPLLE'
''PERINTEISEEN'TAPAAN'VOIT'VAIKKA'KERTOA'TERVEI
SET'KAIKILLE'SUOSIKKIASIOILLESI'KUTEN'ESIMERKIK
SI'KISSOILLE'JA'POLKUVENEILLE''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''",255
; kääntö DASMilla:
;   dasm scroll.asm -o scroll.prg
; ohjelman suoritus Commodore 64
; BASIC- kehotteesta: SYS 4096
```

tana ja sisäänrakennetulla BASIC-ohjelmointikielellä varustettuna erityisen suosittu valinta ensimmäiseksi kodin ohjelmointilaitteeksi.

Kääntäjän toiminta on hyvä tarkistaa emulaattorilla, joka on kohdelaitteen toimintaa jäljittelevä ohjelma. Yleensä nykyiset emulaattorit ovat riittävän tarkkoja simuloimaan koko laitteen toiminnot ja oheislaitteet, joten niitä kannattaa käyttää myös helppona testausvälineenä ohjelmalleen. Varoituksen sanana tosin se, että emulaattoreiden toiminta ei aina vastaa alkuperäisen laitteen toimintaa ja voi johtaa loogisten virheiden ja bugien analysoinnissa valheellisiin tunnistuksiin. Näin voi käydä, jos emulaattori on toteutettu naiivisti sivuuttaen tai oikoen alkuperäisen laitteiston toimintaprosessit, jotka monimutkaisuudessaan ovat haastavia jäljitellä virheettömästi. Täten emulaattoreihin ei voi luottaa ainoana testausympäristönä. Erityisesti analogisten artifaktien ja muiden teknisen suunnittelun kautta arvaamattomien konfiguraatioiden jäljittely digitaalisin menetelmin on vaikeaa esimerkiksi äänen tai kuvan tuotannossa.

Grafiikka- ja äänituotanto

Commodore 64:n ja NESin valinta esimerkeiksi ei ole mikään sattuma. Molemmat laitteet ovat äärimmäisen hyviä kehitysalustoja sekä niiden verrannallisen yksinkertaisuuden että monipuolisuuden takia. Kummallakin laitteella voidaan toistaa moniväristä grafiikkaa ja monikanavaista ääntä. C64 ei ole sidottu pelkästään yhteen kuvatarkkuuteen, ja NES voi skrollata pehmeästi neljän näytöllisen edestä grafiikkaa ruudulla. Molemmat laitteet tukevat myös spritejä, joilla voi rakentaa pelihahmoja ja muita liikkuvia kuvaelementtejä. Äänen tuotanto on molemmilla enimmäkseen oskillaattoreihin perustuvaa analogista piirisynteesiä.

Molemmille laitteille on olemassa useita erilaisia grafiikan ja äänen luomiseen ja muokkaamiseen erikoistuneita ohjelmia. Musiikin tekemiseen löytyy muun muassa NerdTracker II ja GoatTracker ja grafiikan käsittelyyn ja formaattimuunnoksiin YY-CHR ja ConGo. Näitä käyttämällä grafiikka- ja äänidatan käsittely ohjelmakoodissa jää vähäiseksi, jättäen monimutkaisten synteesi- ja grafiikkaominaisuuksien toteuttamisen näiden ohjelmien varaan. Kaiken voi myös tietenkin halutessaan käsitellä itse omin työkaluin alusta lähtien!

Kyseiset ohjelmat ovat nimenomaisesti ristiinkehitystyökaluja, joille vaihtoehtona toimivat myös kohdelaitteessa ajetut editorit. NESin tapauksessa sellaisia ei juuri ole, mutta Commodore 64:lle niitä löytyy enemmän kuin muille laitteille. Lisäksi joillekin grafiikkatiloille edelleen paras piirtomuoto on Commodorella ajettu ohjelma.

Grafiikan ja äänien ohjelmointi onnistuu parhaiten hyvän hakuteoksen kera. Commodoren tapauksessa koneen oma manuaali sisältää kaiken tarpeellisen tiedon muistiosoitteista ja näytönohjaimen ja äänipiirin ominaisuuksista. Kattavan näkemyksen laitteiston ominaisuuksista ja niiden tehokkaimmaista hyödyntämisestä saa lukemalla manuaaleja, muita hakuteoksia ja oppaita sekä erityisesti muiden tuottamaa lähdekoodia ja laitteelle soveltuvan algoritmiikan tutkimusta.

```
// NES-koodiesimerkki: tekstiä ruutuun käyttäen taustagrafiikkamerkkejä.
// mukautettu shirun esimerkistä.

// Tämä esimerkki olettaa, että fontin merkit ovat ASCII-järjestyksessä
// taustagrafiikan tämänhetkisessä merkkisivussa (nametable) kohdissa $00-$3f

#include "neslib.h" // NESLIB mukaan

// makro nametable-osoitteen määrittämiseksi
#define NTADR(x,y) ((0x2000|((y)<<5)|x))

// funktio: kirjoita merkkijono str näyttömuistiin kohtaan adr
void put_str(unsigned int adr,const char *str)
{
   vram_adr(adr);
   while(1)
   {
       if(!*str) break;
       vram_put(*str++)-0x20); // -0x20 = ASCII-koodista merkin numeroksi
   }
}

void main(void)
{
   pal_col(1,0x30); // asetetaan tekstin väri
   put_str(NTADR(2,2),"HEI KAIKKI SKROLLIN LUKIJAT");
   ppu_on_all(); // näytönpiirto päälle
   while(1); // ikiluuppi
}
```

Suoritus

Assemblerin tuottama prg-tiedosto voidaan suorittaa sellaisenaan Commodore 64:llä monella eri tapaa. Tässä muutama, joka soveltuu myös useille erilaisille saman aikakauden kotitietokoneille:

- Ohjelmabinääritiedosto voidaan muuntaa ääneksi, joka tallennetaan kasetille ja ladataan siltä.
- Ohjelma tallennetaan levykkeelle joko käyttäen PC-yhteensopivaa levykelukijaa tai -yhteyskaapelia (esimerkiksi Star Commander ja X1541-kaapeli).
- Ohjelma ladataan suoraan muistiin esimerkiksi C2N232:ta käyttäen.
- Modeemi- tai nollamodeemikaapelisiirto.

NESillä ROM-kuvatiedoston suorittaminen vaatii ROM-moduulien toimintaa jäljittelevän kehityslaitteen, joka simuloi FPGA-piirillä moduulien erikoispiirejä. Laite lukee tiedostot esimerkiksi SD- tai MMC-muistikortilta. Tälläinen *flashcart* on mm. NES PowerPak.

Toinen suosittu tapa on ns. repropakin valmistaminen. Tässä menetelmässä samantyyppisen laitteiston sisältävän pelimoduulin ROM-piirit vaihdetaan itse kirjoitettuihin piireihin, jotka sisältävät oman ohjelman grafiikat ja ohjelmakoodin omilla piireillään. Tämä vaatii kuitenkin jo sen verran elektroniikkaosaamista, että flashcartin hankinta hinnasta huolimatta tekee tästä vaiheesta NESin ohjelmistokehitystä huomattavasti helpompaa.

Esimerkkiohjelmamme NESille on tehty käyttäen suosittua 6502-prosessorille tarkoitettua C-kielen ristiinkääntäjää cc65, joka on saatavilla usealle eri modernille käyttöjärjestelmälle. NES-kehitystä helpottaa huomattavasti neslib-kirjasto, joka tarjoaa loogisesti nimetyn ja jäsennellyn rajapinnan NESin muistiin ja laitteistoon. Paketista löytyy myös monipuolisempia ohjelmaesimerkkejä.

Dreamcast

Samaa lähestymistapaa ohjelmien kehittämiseen voidaan käyttää myös uudemmalla kotikonsolilla, jolle pääsy on ollut suljettu kotikehittäjiltä suurimman osan konsolin elinajasta markkinoilla. Esimerkkimme on Segan Dreamcast-konsoli sen CD-formaatin hankintahelppouden ja laitteen grafiikkaominaisuuksien ja kehitystyökalujen nykyisen saatavuuden takia. Dreamcastin GD-ROM-standardi ei estä mitenkään tavanomaisten poltettujen CD-R-levyjen käyttöä omien ohjelmien säilyttämiseen, kunhan konsoli on vain joko ensin modifioitu lukemaan poltettuja levyjä tai käynnistetty käyttäen erityistä käynnistyslevyä, joka mahdollistaa poltettujen levyjen käytön ohjelmien lataukseen. Yksi suosituimmista on Utopia bootdisk.

Avoimien työkalujen hyödyntäminen

Dreamcastille ohjelmien kehittäminen on helppoa, sillä sen SH4-prosessorille voi kääntää C-kielistä koodia suositulla GNU C-kääntäjällä, gcc:llä. Koska Dreamcastin grafiikka- ja ääniominaisuudet ovat huomattavasti monimutkaisemmat kuin 8-bittisillä konsoleilla, on myös apukirjastojen käyttäminen nopean ohjelmistokehitys- ja testausprosessin helpottamiseksi korkean tason ohjelmointikielen kanssa mahdollinen valinta. Suosittu kaupallisissakin julkaisuissa käytetty vaihtoehto on KallistiOS, joka tarjoaa käyttöjärjestelmäpalveluita ja muita multimediaominaisuuksien ja syötetiedon käsittelyyn soveltuvia kirjastokokonaisuuksia. KallistiOS:llä voidaan kehittää ohjelmia myös Gameboy Advance ja PlayStation 2 -pelikonsoleille.

Grafiikan piirtämiseen voidaan käyttää OpenGL-grafiikkakirjastoa, joka on yhä nykyäänkin suosittu erityisesti siirrettävien eli helposti monelle alustalle käännettävien ohjelmien kehittämisessä. Tässä mielessä Dreamcast-ohjelmien kehittäminen ei merkittävästi eroa nykyisille laitteille kehittämisestä kuin suoritustehonsa ja joidenkin erityisominaisuuksiensa osalta. Dreamcastin etuna on se, että jokainen konsoli on täysin identtinen, ja siten sama ohjelma toimii varmasti jokaisella konsolilla samalla tavalla, lukuunottamatta televisiojärjestelmien virkistystaajuuksien eroavaisuuksia.

Yleiskatsauksena siis kaikki tarvittavat ohjelma- ja laitteisto-osat ovat ilmaisia, helposti saatavilla ja standardinmukaisia, vaikka laitteelle kehittäminen ei "laillisia" teitä käyttäen olekaan mahdollista. Lopullinen käännetty ohjelma syötetiedostoineen tarvitsee vain kirjoittaa ISO-standardin mukaiseen CD-levykuvaan, polttaa fyysiselle levylle sopivin ohjein ja käynnistää modifioidulla tai käynnistyslevyllä varustetulla konsolilla.

Dreamcastille kehittäminen ei ole merkittävästi erilaista kuin nykytietokoneille. On kuitenkin syytä muistaa, että konsoleiden suunnitteluratkaisuista johtuen kaikki ohjelmat, jotka toimisivat moitteettomasti nykytietokoneilla, eivät kuitenkaan toimi Dreamcastillä laitteen huomattavasti pienemmän suoritustehon vuoksi. Laitteen rautaominaisuuksien tehokas hyödyntäminen vaatii yrityksen ja erehdyksen mukanaan tuomaa kokemusta, tietolähteistä lisäsisältöä ammentaen.

Ristiinkehitys vaatii omanlaistaan tutkimusmatkailuhenkeä, sillä saatavilla ei yleensä ole nykylaitteiden laajamittaista tieto- ja tukiverkostoa. Osittain tästä syystä monet vanhat laitteet synnyttävätkin ympärilleen omia harrastelijaryhmiään ja -yhteisöjään. Juuri näistä paras tietämys koneiden syvälliseen sielunelämään usein löytyykin. Tutkimaton potentiaali piilee laitteissa, jotka vasta nyt ovat helposti avattavissa kehitysalustoiksi. Toisaalta vanhoista läpikotaisin tongituista ohjelmoijien suosikkikoneista voi löytyä monia yllättäviä aarteita. Jokainen uusi vuosi tuo tullessaan ennalta arvaamattomilla tavoilla rajoja rikkovia ohjelmia laitteisiin, jotka moni on jo jättänyt menneisyyteen.

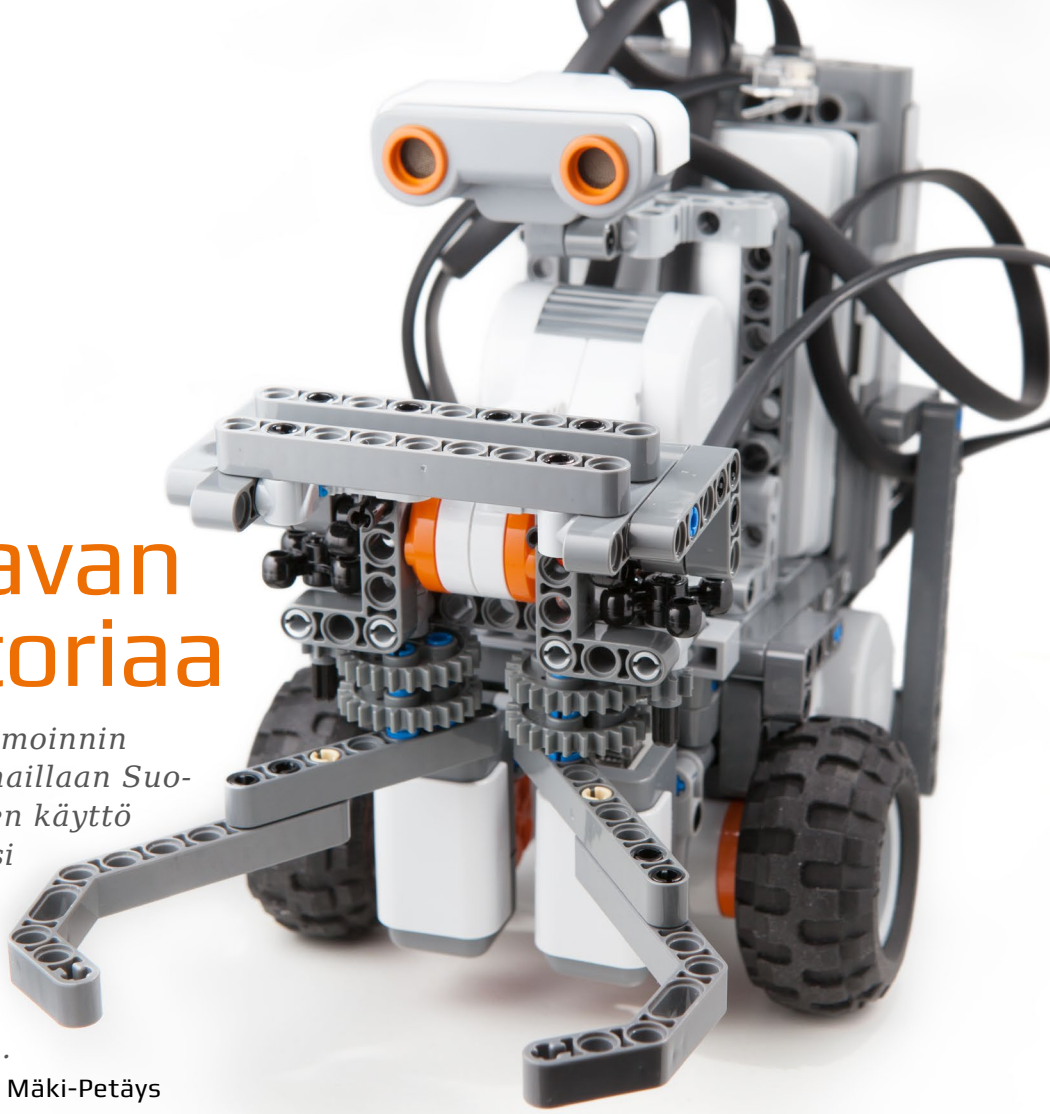
```
// KallistiOS-esimerkkiohjelma. Ei juuri
// eroa perinteisestä C-esimerkistä.

#include <kos.h>

// Oletusasetukset käynnistettäessä
KOS_INIT_FLAGS(INIT_DEFAULT);

int main(int argc, char **argv) {
    printf("\nTerkut konsolistasi!\n\n");
    return 0;
}
```

Dreamcast-kehitysympäristö
http://www.elliptique.net/wiki/doku.php?id=dreamcast



Ohjelmoitavan Legon historiaa

LEGO-palikoiden käyttö ohjelmoinnin opetusvälineenä yleistyy parhaillaan Suomen kouluissa. Robottisarjojen käyttö opetustarkoituksiin ei ole uusi idea: ensimmäiset ohjelmoitavat LEGOt julkaistiin jo 1986. Skrolli tutustuu LEGO-ohjelmoinnin historiaan ja käytännön sovelluksiin.

Teksti: Tomi Pieviläinen Kuvat: Risto Mäki-Petäys

Ensimmäinen ohjelmoitava LEGO oli vuonna julkaistu 1986 LEGO TC Logo™, joka käytti Seymour Papertin Massachusetts Institute of Technologyn Media Labissa lasten ja nuorten opetukseen kehittämää Logo-kieltä. Se pohjautui LCSI:n LogoWriteriin, joka oli jo aiemmin levinnyt koulumaailmaan myös Suomessa. Tuotteen toi markkinoille kouluille tarkoitettuja rakennussarjoja valmistava, kuusi vuotta aiemmin perustettu LEGOn Institutional Division.

Logoa käytettiin myös vuonna 1993 julkaistussa Control Labissa. Sen suosio kasvoi tasaisesti. Control Labeja tuli paikoitellen myös Suomen kouluihin, joissa ne tunnettiin myös ID:n uudella nimellä LEGO Dactana. Niiden käyttöä opetuksessa tutkittiin Oulun yliopistossa 90-luvun loppupuolella.

Control Labin rajoitteena oli jatkuva langallinen yhteys tietokoneen sarjaporttiin. Sillä ei voinut tehdä itsenäisesti liikkuvia mekanismeja vaan kiinteitä järjestelmiä, kuten hissejä tai liikennevaloja. Lisäksi monet opettajat kokivat ohjelmointiympäristön liian vaikeaksi. Tekstipohjainen ympäristö oli herkkä kirjoitusvirheille, eikä ympäristö ollut tarpeeksi kehittynyt auttaakseen aloittelevia ohjelmoijia muistamaan oikeita käskyjä.

Liikkuvien legorobottien kehitys oli aloitettu MIT:ssä jo 1987, jolloin ohjelmoitava legopalikka, The 6502 Programmable Brick, näki päivänvalon. Jatkoprojektit huipentuivat, kun Model 120, The Red Brick, valmistui 1994.

Tie kehittelystä tuotteeksi

Model 120 oli epäkaupallinen prototyyppi, mutta näytti ulkoisesti varsin valmiilta tuotteelta. Sen ominaisuudet olivat samankaltaisia kuin nykyisissä LEGOn älypalikoissa. Punatiileen sai kiinni kuusi anturia ja neljä moottoria. Virran näille antoi sisäinen akku. Kontrolleina toimivat infrapunavastaanotin, kaksi nappia ja nuppi, joita täydensi pieni lcd-näyttö ja kaiutin. Logotulkkia pyöritti Motorola 6811 -prosessori varustettuna 32 kilotavun keskusmuistilla.

MIT:n prototyypit eivät kuitenkaan päässeet myyntiin asti. LEGO kehitti kotikäyttäjille Logon tilalle graafisen RCX Code -ohjelmistoympäristön, ja yhdessä Tufts Universityn kanssa National Instruments LabViewiin pohjautuvan, kouluille tarkoitetun RoboLabin.

Vuonna 1998 RCX ja RoboLab julkaistiin uuden robottialustan kanssa. Kaupalliseksi nimeksi tuli Mindstorms: Robotics Invention System, kouluille tarkoitetusta puolestaan RoboLab: Mindstorms for Schools. Sarja tunnetaan myös ikonisen keltaisen Robotic Command eXplorer -keskusyksikön, eli RCX:n nimellä.

RCX oli suunnittelultaan hyvin samankaltainen MIT:n mallin kanssa, joskin hieman rajoitetumpi. Siihen sai kytkettyä kolme moottoria ja kolme anturia. Kontrolleina toimivat infrapunavastaanotin ja neljä nappulaa. Yksikössä oli LCD-näyttö, mutta ei akkua. Se käytti kuutta AA-paristoa, joskin ensimmäisiin versioihin sai myös ulkoisen virtalähteen. Vaikka yhteistyö MIT:n kanssa oli päättynyt, LEGO nimesi sarjan kunnianosoituksena Mindstormsiksi Seymour Papertin samannimisen kirjan mukaan.

Ensimmäinen sarjatuotettu ohjelmoitava LEGO oli valtava menestys ja sille tehtiin paljon epävirallisia firmwareja. Lisäksi RCX:lle tehtiin ohjelmistoympäristöjä lukuisille kielille, esimerkiksi C:lle, Javalle ja Matlabille.

Suomen kouluihin LEGO-robotiikkaa

RCX:n ensimmäisiä käyttäjiä Suomessa oli Oulun yliopistosta Exeterin yliopistoon siirtynyt Esa-Matti Järvinen. Hänet palkattiin 2001 Ylivieskan

AMK:n perustamaan teknologiakeskus Teknokkaaseen. Jo vuonna 2002 järjestettiin Oulussa Suomen Tekoälyseuran, VTT Elektroniikan ja Oulun yliopiston voimin ensimmäiset Robocup Junior -kisat, jossa lajeina olivat jalkapallo, pelastus ja tanssi.

Seuraavina vuosina kisat järjestettiin Heurekassa ja uudelleen Oulussa, ja ne vakiintuivat jokavuotisiksi. RCJ:n päätyminen yleisimmäksi kilpailutapahtumaksi oli Pohjois-Euroopassa harvinaista, sillä LEGO järjestää myös omaa First LEGO League -tapahtumaansa. Esimerkiksi Ruotsissa ja Virossa FLL on saanut suuren suosion.

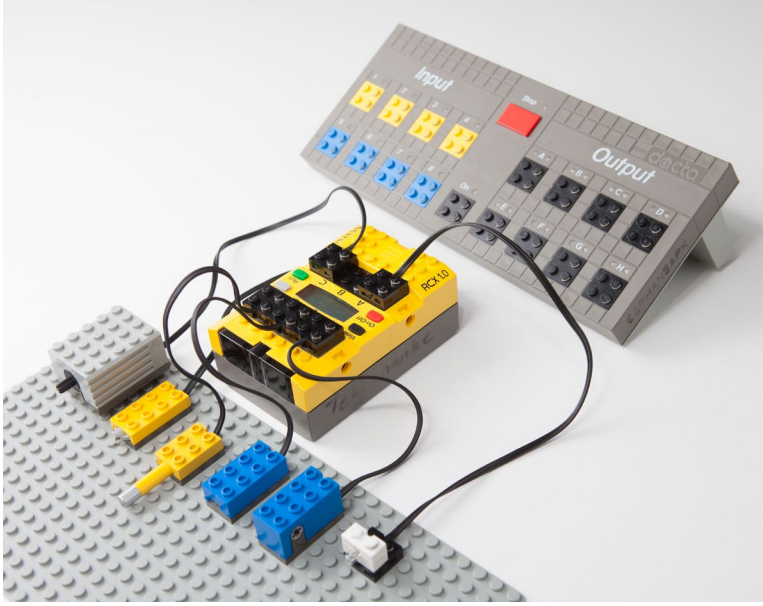
RCX uusittiin täysin vuonna 2006, kun Mindstorms NXT julkaistiin. Sekä kaupalliset että kouluversiot käyttivät uutta graafista NXT-G-ympäristöä, joka pohjautui vieläkin LabViewiin, mutta ei ollut yhteensopiva RoboLabin kanssa. Sarja muuttui ulkoisesti suuresti, sillä uudessa sarjassa käytetään pääasiassa nappulattomia, moderneista tekniikkalegoista tuttuja palkkeja. Myös kaikki sähköiset osat kiinnitettiin uudella liitännällä, joskin kouluversion mukana toimitettiin adaptereita vanhojen RCX-osien käyttämiseksi.

NXT:n suosio näkyy sensoreiden tarjonnassa. Niitä voi ostaa LEGOn lisäksi kolmelta muultakin yritykseltä. Yksinkertaisimmat näyttävät ulkopuolisesti virallisilta osilta, mutta sisältävät kalliimpia sensoreita, kuten kompassin tai kiihtyvyyssensorin. Mutta tarjolla on myös osia LEGOn maailman ja yleisen elektroniikan rajapinnan hämärtämiseksi, kuten adaptereita LEGOn kaapeleille ja tavalliselle koekytkentälaudalle. Koska NXT:n liittimien tiedot ovat julkisia, niiden avulla kuka tahansa voi liittää keskusyksikköön oman analogisen tai I2C:tä käyttävän sensorinsa.

RCX oli viimeisenä vuotenaan ehtinyt levitä myös korkeakouluopetuksen puolelle. Turun yliopistossa LEGOt oli otettu käyttöön yhteistyöprojektissa Uppsalan yliopiston kanssa. Kurssilla loppuvaiheen opiskelijat suunnittelevat Internetin läpi käytettävän robotin, joka pystyy tutkimaan tuntematonta aluetta yhteistyössä toisen robotin kanssa. Esikuvana toimivat tälläkin hetkellä Marsin pintaa kartoittavat robotit. Hyvien kokemuksien jälkeen Turussa LEGOt on otettu käyttöön myös fukseille insinöörityöhön johdattavalla kurssilla.

Myös Helsingin Yliopisto opettaa fukseille legoilla ohjelmoinnin perusteita, ja käytäntö on levinnyt sieltä myös Metropoliaan. Valkoisiin NXT-robotteihin voi törmätä myös Tampereen ja Lappeenrannan teknillisillä yliopistoilla, jossa sulautettujen järjestelmien kursseilla liikkuvat laitteet on huomattu motivoivemmiksi kuin perinteiset kehitysalustat.

Tämä motivoiva vaikutus on myös keskeisimpiä syitä, miksi Opetushallitus on rahoittanut parina viime vuonna LEGOjen hankintaan kymmeniin eri peruskouluihin. Robotiikan opetus on vaatinut aiemmin hyvin suurta kiinnostusta opettajilta, sekä onnea oikeiden kontaktien suhteen. Nyt uutta teknologiaa tuodaan kuitenkin opetukseen mukaan määrätietoisesti.



LEGO Mindstorms RCX -sarjan DACTA-hallintapaneeli, ohjainyksikkö, moottori, kosketus-, lämpötila-, valo- ja kulmasensori sekä valo

Uutta on yhä luvassa

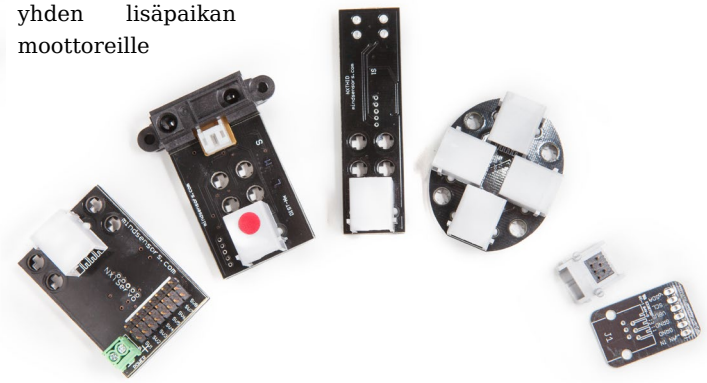
LEGOkaan ei lepää rattailla. Tammikuussa CES-messuilla esiteltiin uusi versio Mindstormsista: EV3. Ulkoisesti se muistuttaa NXT:tä hyvin pitkälti, mutta sisus on päivittynyt reippaasti. Sekä muistia että prosessointitehoa on moninkertaisesti, joten EV3:n sisällä pyörii täysiverinen Linux-järjestelmä.

Ohjelmointiympäristöjen mahdollisuudet laajenevat valtavasti, kun vanhemman sulautetun kehitysympäristön sijaan mitä tahansa ARMille siirrettyä Linux-ohjelmaa voidaan käyttää suoraan. EV3 sisältää microSD-muistipaikan, yhden lisäpaikan moottoreille ja USB-portin, johon voi kytkeä vaikka wlan-mokkulan.

Uusia sensoreita ovat ennen vain ulkopuolisten valmistajien tekemät gyroskooppi ja IR-lähetinvastaanotin. Myös kaikki vanhat sensorit toimivat myös uudessa keskusyksikössä. EV3:n ohjelmisto seuraa uusia trendejä, ja mukana tulee suoraan LEGOn viralliset Android ja iOS-sovellukset. Itse rakennusohjeetkin ovat jatkossa paperin lisäksi saatavana myös tabletilla pyöriteltävinä 3D-malleina. EV3:n kouluversio tulee myyntiin elokuussa ja kotiversio myöhemmin tänä syksynä.

Juttua varten haastateltiin Erkki Hautalaa Espoon Vanttilan koulusta, Petteri Kartimoa Rovaniemen Saaren koulusta, Arto Karsikasta Jokilaakson koulutuskuntayhtymästä, Tero Ahosta Lappeenrannan teknillisestä yliopistosta, Seppo Virtasta ja Ville Taajamaa Turun yliopistosta ja Simo Silanderia Metropoliasta.


LEGO Mindstorms NXT -sarjan moottori, valo- ja äänisensori, ultraäänietäisyysmittari, kosketus- ja kiihtyvyyssensori sekä ohjainyksikkö


Mindsensors.com:in NXT-moduuleja: servo-ohjain, infrapunaetäisyysmittari, USB-HID-muunnin, porttijakaja ja koekytkentälevyn liitin

Dwarf Fortress
Linnoitusta perustamaan

Jos tavallisten kuolevaisten pelit ovat sinulle liian yksinkertaisia, voit haluta tutustua Dwarf Fortressiin. Pidä tosin varasi. Jättimäisen adamantiumpiikeillä koristellun teräskynnyksen yli yltävät jäävät usein täysin koukkuun.

Teksti: Kalle Viiri, Tapio Berschewsky
Kuvat: Tapio Berschewsky

Bay 12 Gamesin Dwarf Fortress -peli on hyvin erikoinen kokonaisuus. Se on rakentelupeli, jossa veri roiskuu ja kissoista tehdään saippuaa. Se on roguelike, jossa ylitetään valtameriä uimalla ja kuristetaan rosvoja elävältä. Se on sekä strategia- että seikkailupeli laajassa satunnaisgeneroidussa maailmassa, jossa saattaa vaeltaa suolasta tehtyjä titaaneja ja luonnottomia yömörköjä. Jos haluat leikkikentäksesi fantasiamaailman linnoineen, sankareineen, villipetoineen ja mielikuvituksen rajoja koettelevine kummajaishirviöineen, Dwarf Fortress on peli juuri sinulle.

Dwarf Fortress alkaa maailman luomisella, joka voi kestää pitkän tovin. Peli pyrkii luomaan realistisen maailman simuloimalla erilaisia luonnonvoimia ja niiden vaikutusta maastoon. Kun maasto on valmis, simuloidaan käyttäjän valintojen pituinen ajanjakso maailmaan luotujen kansojen ja julmien hirviöiden keskinäisiä kamppailuja. Sivilisaatiot nousevat, leviävät, sotivat, ja tuhoutuvat. Kun prosessi on valmis, pelaajalla on edessään kokonainen maailma täynnä tutkittavaa. Kaupunkien alaisiin katakombeihin kätketyt aarteet odottavat ottajaansa ja aikojen alusta maan päällä vaeltaneet titaanit surmaajaansa!

Muurinpala kerrallaan

Kun pelimaailma on luotu, pelaaja päästetään osaksi sitä. Pelimuotoja on kaksi: Fortress Mode ja Adventurer Mode. Suositumpi on Fortress Mode, jossa pelaaja pääsee ohjaamaan pientä kääpiöretkikuntaa uuden linnoituksen rakentamisessa. Tämä on se pelimoodi, johon yleensä viitataan Dwarf Fortressina. Adventurer Mode puolestaan muistuttaa enemmän perinteistä roguelikea, jossa pelaaja ohjaa yksittäistä hahmoa seikkailemassa ympäri pelimaailmaa.

Fortress Mode alkaa pelaajan valittua sopivan paikan uudelle kääpiölinnoitukselleen. Retkikunnan seitsemän jäsenen kyvyt ja varusteet voi viilata viimeisen päälle itselleen sopiviksi tai käyttää vältävän hyviä oletusasetuksia. Retkikunnan saapuessa uudelle valtaukselleen kääpiöt voi käskyttää oitis toimeen.

Lyhyet veijarit tulevat toimeen tarvittaessa vähällä. Ennen pitkää pelaajan on kuitenkin huolehdittava linnoituksen asukkien perustarpeista, jotta projekti ei kariutuisi heti alkuunsa. Kääpiöille voi makunsa mukaan rakentaa saatavilla olevista materiaaleista asuintiloja, työpajoja ja varastoja maan päälle tai louhia tarvittavat tilat kallioon. Pelimaailman kolmiulotteisuus ja kääpiöiden legendaariset rakennustaidot antavat pelaajan luovuudelle paljon tilaa linnoituksen suunnittelussa.

Kääpiöt eivät tee mitään omin päin, eikä pelaaja ohjaa pienokaisia suoraan. Varsinainen käskytys tapahtuu suunnittelemalla se, mitä pitäisi tehdä, minkä jälkeen sopiviin ammatteihin määrätyt hahmot alkavat toteuttamaan sitä mahdollisuuksien mukaan.

Runsaudensarvellinen sälää

Tilojen rakentamisen ja louhimisen ohella peli tarjoaa kiitettävän monia erilaisia askareita kääpiöiden iloksi. Kääpiöt osaavat hankkia ravintoa linnoitukseen viljelemällä maata, metsästämällä, keräämällä luonnonvaraisia kasveja, kalastamalla ja hoitamalla karjaa tai mehiläisiä. Kääpiöille hyvin maistuva alkoholi syntyy kasveista tai hunajasta.

Uusia vaatteita ja säkkejä kuluneiden tilalle voi valmistaa eläinten villasta, nahasta, luolassa elävien hämähäkkien seitistä valmistetusta silkistä tai kasvikuiduista. Huonekalut ja useimmat käyttöesineet syntyvät pääasiassa puusta, metallista tai kivestä. Erilaisia koriste-esineitä voi valmistaa näiden lisäksi myös eläinten luusta, savesta, hiekasta poltetusta lasista ja maan alta löydetyistä jalokivistä.

Oravannahat ja teräskirveet

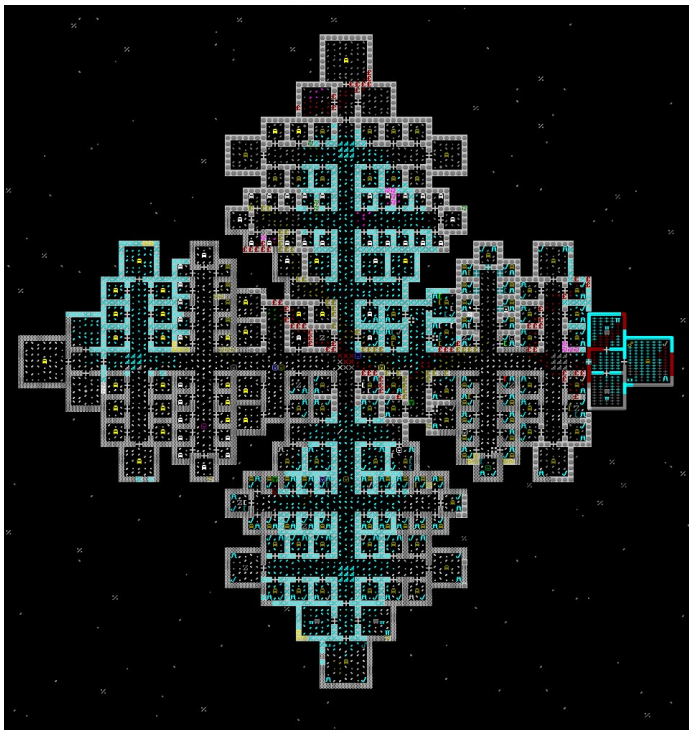
Kääpiöt ovat ahkeria veijareita. Pelin alussa käskytettävillä

Mikä ihmeen roguelike?

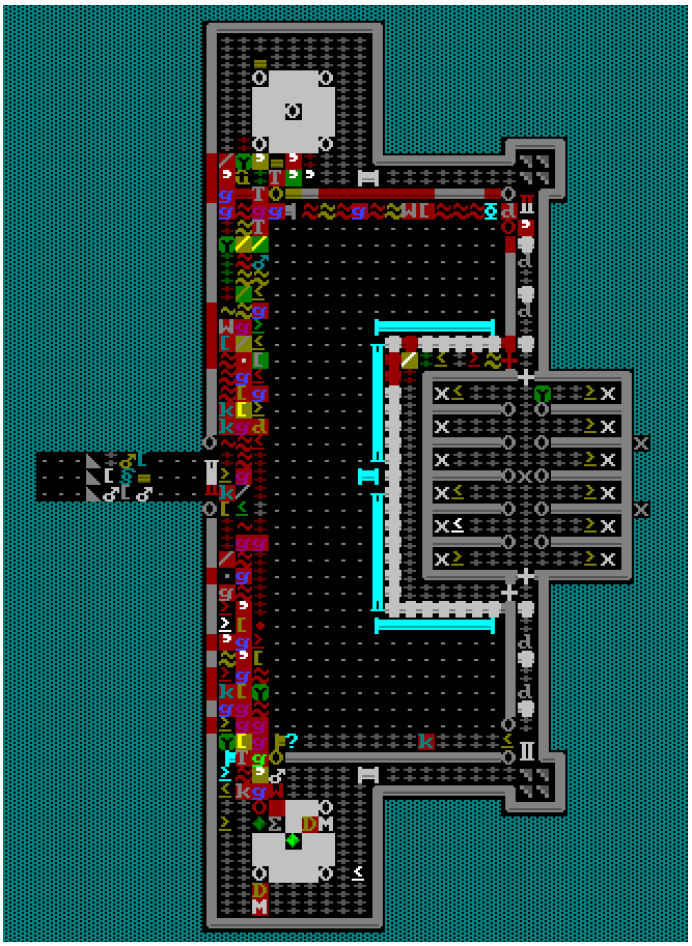
Jäikö ymmärtämättä, mitä tarkoitetaan rosmonkaltaisella? Termi roguelike viittaa 80-luvulla suositun merkkigraafisen Rogue-pelin henkisiin seuraajiin. Pelit tarjoavat ympäristöksi satunnaisgeneroituja maailmoja, jotka ovat joka pelikerralla erilaisia. Tyypillisesti roguelike-termiin mielletään myös grafiikkatyyli, joka koostuu kuvien sijasta kirjaimista ja muista merkeistä. Tämä ei kuitenkaan ole oleellinen osa termin merkitystä, ja monia roguelikejä voi pelata myös graafisissa moodeissa.

Tunnetuin roguelike kautta aikojen lienee itse Roguen lisäksi huippusuosittu NetHack, joka on yhä yksi maailman monimuotoisimmista ja hienoimmista peleistä. Muita merkittäviä ovat esimerkiksi klassinen Omega, laaja ja hieno ADOM, sekä nykyään yleisesti ykkössijalle nostettu Dungeon Crawl Stone Soup. Rogueliket ovat pitkälti ilmaisia, ja niitä voi yleensä pelata verkossa samalla palvelimella muiden kanssa. Vaikka pelit ovat yksinpelejä, pääsee näin jakamaan pistesijalistan. Kilpailu siivittää myös roguelikeissä parempiin suorituksiin.

NetHackia voi kokeilla verkossa esimerkiksi ottamalla telnet-yhteyden osoitteeseen nethack.alt.org.



Jokaiselle pienelle viinanhimoiselle penteleelle pitää olla oma tönö. Jehuille tulee isommat. Näitä on kiva väkertää ja sisustaa. Paitsi jos ei tykkää.



Neljänteen kerrokseen maan pinnalta nostettu porttirakennus. Sisään tullaan lännestä keskeltä. Pohjoisessa ja etelässä sijaitsevat kauppapaikat, oikealla jousimiesten harjoitusalueet, sekä turkoosin laskusillan takana parveke, josta ikävät sisääntulijat voi teurastaa ilman lähikontaktia. Koko komeuden jokainen reitti voidaan sulkea ja avata yksitellen vivuista 20 kerrosta alempana. Koko vasen kulkuväylä on täynnä vihollisten raatoja, ruumiin kappaleita, oksennusta ja siihen sekoittuvaa verta.

seitsemällä kääpiöllä on yllin kyllin energiaa tarvitsemiensa asumusten, viljelmien, huonekalujen ja työvälineiden tekoon, ja käsitöitä jää helposti ylikin. Tällöin naapurikansojen liikemiesvaistot heräävät, ja pelaaja pääsee käymään kauppaa ihmisten, haltioiden sekä oman kääpiökansansa pääkaupungin kanssa.

Kauppakumppanien käyttäytyminen ja tarjottavat tuotteet riippuvat rodusta. Haltiat kauppaavat lähinnä puusta tehtyjä tuotteita, itse tekemiään ruokia ja juomia sekä harvinaisia eläimiä. Nipukkakorvaiset ituhipit ovat tosin oikukkaita kauppakavereita. He saattavat jopa julistaa sodan, jos heille tarjotaan vastineeksi esineitä, joiden valmistuksessa on käytetty puita. Ihmiset tarjoavat ja hyväksyvät monipuolisempaa kauppatavaraa, mutta kääpiöiden karavaani tuo yleensä rikkaimmat lastit ja harvinaisimmat metallit. Mikäli kääpiösivilisaation käytössä on sopivia resursseja, he tuovat myös korkealuokkaista ja kallista terästä, jota muut kansat eivät osaa valmistaa.

Hiljalleen viennillä ja teollisuudella vaurastuva linnoitus houkuttelee siirtolaisia kääpiövaltion eri osista. Tarpeeksi maallista mammonaa ja asukkaita keräämällä pelaajan linnoituksesta voi tulla paroni-, kreivi- tai herttuakunta sekä lopulta koko kääpiövaltion pääkaupunki. Samalla pelin vaikeustaso nousee. Aateliset tekevät mielivaltaisia vaatimuksiaan esineiden valmistamisesta ja viennistä ja tuomitsevat alempiarvoisia ilomielin vankeuteen jos määräyksiä ei täytetä.

Ei pelkkää leikkiä

Jännitystä vaurastuvan linnoituksen elämään tuovat kääpiömaailman vaarat. Linnoituksen alueella normaalisti hyörivien villieläinten ja mahdollisten epäkuolleiden lisäksi päänvaivaa aiheuttavat kääpiöiden omaisuutta kähveltävät kelmit ja lapsia kaappaavat ryövärit. Linnoituksen kasvaessa porteille voi marssia kokonainen armeija jättiläissammakoilla tai hirviölinnuilla ratsastavia häikäilemättömiä menninkäisiä.

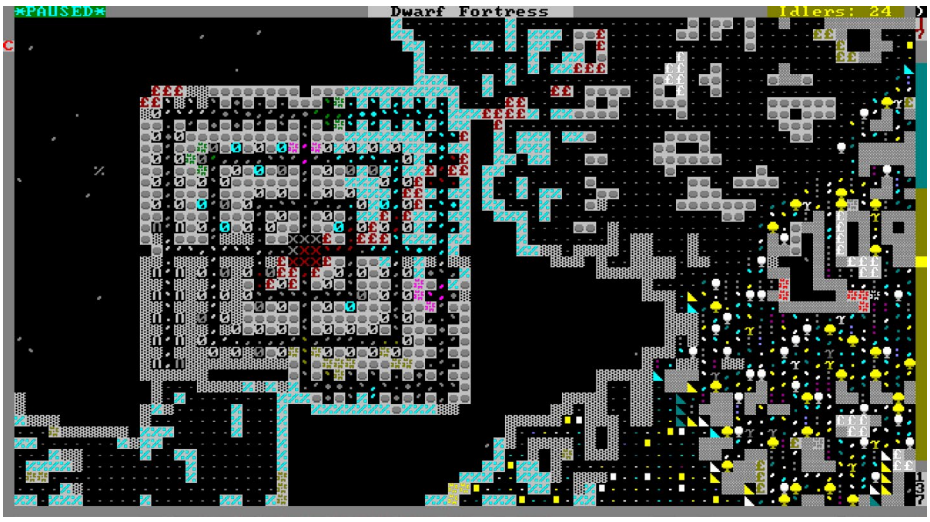
Harvinaisempia hyökkääjiä ovat maan päällä kiertelevät titaanit ja muut jättihirviöt. Kovimman vastuksen antavat kuitenkin maan alla luolastoissa vaanivat muinaiset ja unohdetut satunnaisluodut megapedot, jotka vain odottavat tilaisuutta päästä mellastamaan varomattomasti kaivavien kääpiöiden linnoituksiin. Kun vihollinen on ikiaikainen vesihöyrystä koostuva demoni, joka sylkee myrkyllistä laavaa kymmenien metrien päähän, ei teräksellä enää tee juuri mitään – väliin tarvitaan kiveä.

Pelaajalle tarjotaan puolustusta varten laaja valikoima erilaisia tapoja selättää vihollisuhat. Yksinkertaisinta on pystyttää muuri, mutta koska kauppa ja maahanmuuttajat vaativat reittiä linnakkeen sydämeen, se ei kelpaa ainonana ratkaisuna kuin ensimmäisiin linnoituksiin.

Kääpiöt ovat taitavia virittämään ansoja, kuten terässirkkeleitä, elävänä kiinni ottavia häkkejä, putoavia kattoja, murskaimia ja uskomattoman syviä teräspiikeillä koristeltuja kuoppia. Ahkerampi pelaaja voi jopa käskyttää kääpiönsä rakentamaan pumppujärjestelmän, joka tuo magmaa suoraan maan ytimestä vihollisarmeijan niskaan.

Useita villieläimiä voi pyydystää ja kouluttaa vartioimaan linnoitusta. Mikäs sen mukavampi kuin oma lohikäärmelauma raatelemassa kutsumattomia vieraita hengiltä. Kääpiöt ovat itsekin urheita sotureita niin lähitaistelussa kuin kauempaa varsijousella ampuen. He osaavat myös käyttää katapultteja ja ballistoja vihollisarmeijoiden rivien murtamiseen.

Taisteleminen itsessään on niin brutaalia, että valiojoukkoja kannattaa kuitenkin säästellä. Kun kymmenen vuotta treenannut huipputason kaivosmies ottaa iskun selkäruotoon ja halvaantuu, ei enää juuri huvita, vaikka



Viimeisiä leposijoja tarvitaan paljon, sillä kääpiöt ovat helposti rusentuvaa sakkia. Kuolevat usein ihan tyhmyyttään. Symmetrisen hautausmaan lisäksi kartalla näkyy paljastunutta luonnon muovaamaa luolastoa.

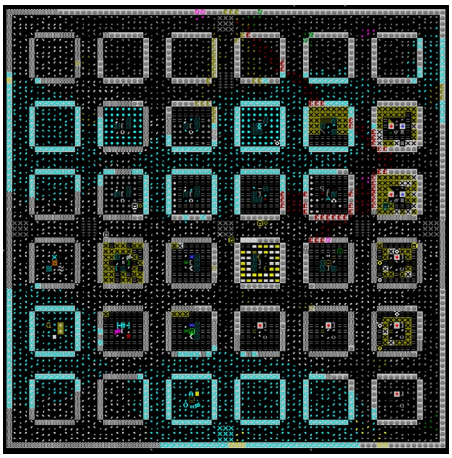
pihalla olisikin kymmeniä kasoja vihollisruumiita sankaruuden todisteena.

Päältä kaunis, sisältä mätä

Linnoituksen puolustaminen ulkoisilta uhilta on kokemuksen myötä suhteellisen helppoa, mutta lisähaasteena kääpiöt ovat myös mestareita tekemään toistensa elämästä jatkuvaa piinaa. Jokaisen kääpiön mieliala voi laskea erikoisista syistä. Ehkä viina oli loppu kun kääpiölle iski jano, tai kenties hampaankolossa kaihertaa naapurin hienompi asunto.

Tarpeeksi ikäviä asioita sietämään joutunut kääpiö voi ryhtyä mellakoimaan ja rikkomaan paikkoja, mistä voi seurata lisää riehuvia kääpiöitä ja heiluvia nyrkkejä. Ja nämähän muistetaan. Jonain päivänä Urist McRaivomieli muistaa masennuksen alhoissaan kuinka serkku veti vaimoa turpaan kymmenen vuoden takaisessa tappelussa viimeisistä viinatipoista. Kirves alkaa viuhua ja pian koko linnoituksen väestö murhaa toisiaan punahehkuisen vihan vallassa.

Tämä fanien keskuudessa "raivarispiraalina" tunnettu ketjureaktio on yksi yleinen lupaavia linnoituksia tuhoava luonnonvoima. Se pakottaa pelaajan pitämään erityisen tarkasti huolta siitä, että kääpiöiden mieliala pysyy aina tarpeeksi korkealla. Mielenrauhaa voi edistää monin tavoin. Tärkeimpiä ilonlähteitä ovat kivat asunnot sekä jättimäinen koristeltu ruokasali. Myös vesiaiheet, eläintarhat, galleriat ja monet muut hienoudet helpottavat sisäistä poltetta pistää kaveria pataan.



Onnellinen työläinen on kuutiossa oleva työläinen. Kaikki pöydät samoin päin, ja ei hymyillä. Linnakkeen erinäiset pajat ja muut tuotantorakennukset varastoineen vievät paljon tilaa.

Salamyhkäisempiä uhkia linnoitukselle ovat siirtolaisten mukana mahdollisesti tulevat verenhimoiset vampyyrit. Kun sieltä täältä alkaa löytyä tyhjiin imettyjä ruumiita, alkaa vaativa salapoliisityö. Kun yön tumma lepakkokääpiö lopulta löytyy, voi oikeuslaitos tuomita tämän murhista kuolemanrangaistukseen. Vähemmän byrokraattinen kääpiö tosin muuraa vampyyrin suosiolla seinän sisään, eikä välitä hienouksista.

Lennän, polskin ja uin

Adventurer Mode antaa pelaajalle mahdollisuuden tutustua pelimaailmaan koko komeudessaan. Toisin kuin Fortress Modessa, pelaaja voi valita hahmokseen myös haltian tai ihmisen. Varsinaista päämäärää Adventurer Modessa ei ole, mutta pelimaailman hahmojen kanssa juttelemalla voi saada vihiä rosvojoukoista, ilkeistä velhoista tai hirviöistä.

Erilaiset uroteot kasvattavat hahmon mainetta, ja tekevät hahmosta elävän osan pelimaailman historiaa. Lisäksi pelaaja voi ryhtyä nekromantikoksi, lykantroopiksi tai vampyyriksi. Seikkailumoodissa voi myös käydä tutustumassa Fortress Modessa rakentamiensa linnoituksien saloihin.

Sekä Adventurer- että Fortress Mode mallintavat pelimaailmaa silmiinpistävän tarkasti. Jokaisesta esineestä ja rakennuksesta voi nähdä, mistä materiaalista se on tehty. Pelimaailmasta löytyvät esineet saattavat paljastaa historiallisia tapahtumia. Esimerkiksi legendaarisen laadukkaisiin sukkiin kudottu koristetaide voi kertoa vuosikymmeniä sitten tapahtuneesta historiallisesta taistelusta.

Myös elävien olentojen taistelussa tai onnettomuuksissa vammautuminen tapahtuu pitkälti reaalimaailman järjen mukaan. Yksittäisen kestomittarin sijaan jokaisen eliön ruumiinosien kuntoa tarkkaillaan erikseen. Taistelussa voi menettää esimerkiksi yksittäisiä sormia,

Ota tämä, tarvitset sitä

No hyvä on, myönnetään sitten. Vaikka Dwarf Fortress on yksi parhaista peleistä ikinä ja varmasti monimutkaisimpia virityksiä, joita pelimaailma on nähnyt, se ei ole aivan helppo pelata. Toisin sanoen sen käyttöliittymä on silkkaa helvettiä, ja vaatiikin koko monimutkaisen komentovalikkorakenteen ulkoa opettelemista lihasmuistiin asti, ennenkuin peli alkaa sujua. Suurin osa kaatuu matkalla eikä yritä ikinä uudelleen.

Aivan täysin tyhjin käsin ei kuitenkaan kannata edes aloittaa. Vaikka alkupäässä peliä pärjää kohtuullisesti oletusliittymällä, varsinkaan linnoituksen kasvaessa ei sen kanssa tule kuin vihaiseksi. Suurin ongelma on kääpiöiden ammattivalintojen määrittely yksi kerrallaan. Tähän on onneksi erinomainen työkalu: Dwarf Therapist.

Therapist näyttää Excel-taulukkoa muistuttavan listan kääpiöistä sekä kunkin yksilön valituista ammateista. Kätevästä ohjelmasta hallitsee vaikka 200 kääpiön linnaketta helposti. Vielä parempi on kuitenkin tästä edelleen kehitelty versio nimeltä Splintermind Attributes. Siinä missä vaniljaterapia mahdollistaa vain ammattivalinnat, Splintermindin versiossa näkee todella paljon muutakin hyödyllistä tietoa kääpiöiden ominaisuuksista, ja esimerkiksi parhaiden kääpiöiden valinta armeijaan helpottuu selvästi.

Erittäin hyödyllinen lisätyökalu on myös df-hack, joka avaa erillisen komentorivi-ikkunan pelin käskyttämiseen. Tätä voi käyttää puhtaasti huijaamiseen, mutta sillä on myös rehellisiä käyttötarkoituksia, jotka helpottavat valikkosumassa turhautumista. Siinä missä Therapist on hyödyllinen heti alkuun, dfhackista saa irti kaiken kunnolla vasta kun on jo tutustunut peliin ja sen konsepteihin.

Keisari neljissä eri vaatteissa. Vasemmalla yläkulmassa pelin mukana tulevan curses-tiilisetin 16x16 pikselin versio. Sen vieressä muutamasta suositusta tiilisetistä henkilökohtaiseen käyttöön sekoitettu versio 16x16 pikselin tiilillä. Tällä setillä on otettu artikkelin muut kuvankaappaukset. Vasemmalla alhaalla on käytössä Yayon tribuutti Commodore 64:n merkistölle. Oikeassa alanurkassa on käytössä vielä kesken oleva uusi 22x22 pikselin tiilisetti, joka julkaistaan joskus sen valmistuessa Skrolli.fi:ssä. Lisää löytyy esimerkiksi Dwarf Fortress -wikin avulla.

murtaa kylkiluunsa tai vuotaa kuiviin kaulavaltimon avanneen iskun seurauksena.

Adventurer Modessa tämä tekee taistelusta hauskaa, sillä iskujen tähtääminen eri ruumiinosiin mahdollistaa tyypillistä roguelikeä taktisemman taistelun ja vihollisten brutaalin murjomisen ennennäkemättömän yksityiskohtaisesti. Jopa niin pitkälle, että kahden käden taistelukirveellä voi osua vaikkapa taaimmaiseen poskihampaaseen ja tehdä siihen pienen loven.

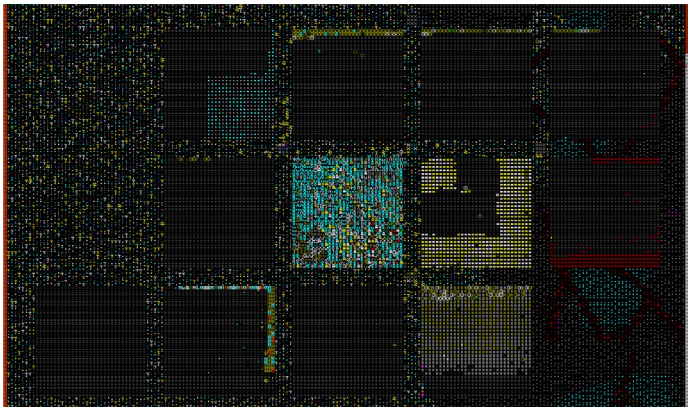
Maailman raskain ruma peli

Pelimaailman yksityiskohtaisuus aiheuttaa myös sen, että peli on ulkoasustaan huolimatta melko vaativa suoritustehon suhteen. Linnoituksen kasvaessa simulointiin kuluu jatkuvasti enemmän aikaa, ja pelaaminen hidastuu nopeammillakin koneilla. Oletusta pienemmän pelialueen valitsemalla ja suunnitelmiaan optimoimalla pelin saa pysymään sutjakkana pidempään.

Vuoropohjaisessa Adventurer Modessa tilanne on sama. Peli reagoi näppäimistökomentoihin selvällä viiveellä, kun pelaajahahmo liikkuu satojen hahmojen asuttamassa kaupungissa allaan valtavia katakombeja ja ympärillään sotilaiden vartioimia torneja. Tämän vuoksi Dwarf Fortressia varten tarvitsee kohtuullisen tehokkaan tietokoneen.

Peli tarjoaa kuitenkin asetustiedoston, jota muokkaamalla voi rajoittaa linnoituksensa väkiluvun kasvua suorituskyvyn parantamiseksi ja linnoituksen hallinnan helpottamiseksi.

Dwarf Fortress on uniikki. Vaikka se on periaatteessa samaa genreä kuin Sims, ja sille on lukuisia kevyitä ja graafisesti vetoavampia klooneja, ei mikään näistä raapaise edes pintaa siitä, mitä Dwarf Fortressilla on tarjota. Jos haluaa pelin, joka palkitsee vuosien ajan joka pelikerralla uusilla asioilla, tätä pidemmälle ei kannata katsoa. Ja muista pelin motto: häviäminen on hauskaa.



Sälää kertyy, joten varaston pitää olla iso. Todella iso. Joskus tekee mieli tyhjentää koko kerros, että saa lisää sälää. Tai miksei parikin.

Opettakaa Tarn Adams koodaamaan

Ville-Matias Heikkilä

Dwarf Fortressin pitäisi periaatteessa olla unelmieni täyttymys. Jo aikoinaan Nethackia ja Omegaa kolutessani kaipasin niihin vieläkin enemmän syvyyttä: yksilöitä ja yhteisöjä, joita simuloitaisiin ruohonjuuritasolta alkaen. Siksi olenkin vuosien mittaan monesti yrittänyt perehtyä Dwarf Fortressiin. Perehtyminen on kuitenkin joka kerta tyssännyt yhteen ja samaan ongelmaan, joka saattaa kuulostaa vähäpätöiseltä, mutta vie joka kerta maun koko pelistä: näytönpäivitys on toteutettu täysin aivokuolleesti.

Vaikka kuva pysyisi muuttumattomana tai koko pelin laittaisi paussille, näyttöä päivittävä silmukka pyörii silti hulluna tyhjää ja tuuletin huutaa. Ohjelmoijaa ei kiinnosta korjata ongelmaa, vaikka se saa hänet vaikuttamaan kädettömältä apinalta. Koska pelin lähdekoodit pimitetään roguelike-perinteiden vastaisesti loppukäyttäjiltä, on itse korjaaminenkin turhan työlästä.

Valitan, mutta en pääse tästä yli. Toivottavasti jo parin vuoden sisällä saadaan pelistä versio, jota ei tarvitse enää myötähävetä heti alkumetreillä. Ennen sitä teen mieluummin itse omat maailmasimulaattorini.

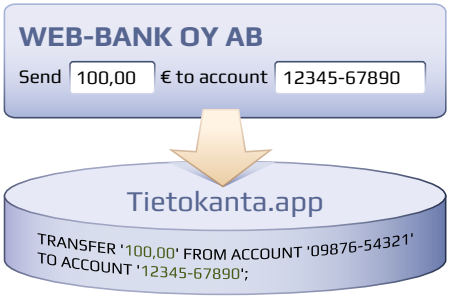
Käytännön tietoturvaa:

SQL-injektio — miten se tehdään ja miten siltä suojaudutaan

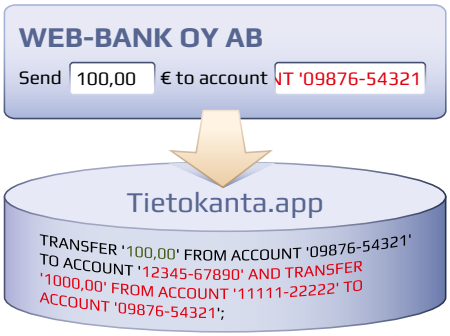
SQL-injektiot ovat eräs vaarallisimmista ja yleisimmistä verkkopalveluita piinaavista tietoturvaongelmista. Palveluiden suojauspuutteista seuraa päivittäin tietovuotoja ja rahallista vahinkoa. Skrolli avaa injektiohyökkäysten anatomiaa monia tietoturva-aukkoja havainnollistavan Damn Vulnerable Web Applicationin avulla.

Kuvat: sxc.hu (Derrick Collins), Risto Mäki-Petäys

Injektiohaavoittuvuudet ovat tavallisia dynaamisissa verkkopalveluissa, niiden löytäminen vaatii vähäistä teknistä vaivannäköä ja niitä on helppo käyttää hyväksi. Tietoturvayhtiö Imperva tutkimusten mukaan tyypillinen verkkopalvelu joutuukin sietämään noin 70 injektioyritystä joka tunti. Useimmissa verkkopalveluissa käytetään yleiseen SQL-kyselykieleen perustuvia relaatiotietokantoja, ja SQL:ään perustuvat injektiot ovatkin verkon yleisimpiä hyökkäystapoja.



Kuva 1: Käyttöliittymään syötetty arvo muunnetaan tietokantalauseeksi.



Kuva 2: Hyökkääjä käyttää hyväkseen huonoa syötteenkäsittelyä ajaakseen ylimääräisen komennon.

Injektiohyökkäyksessä hyökkääjä pyrkii valitsemaan verkkopalvelulle lähetettävät syötteet siten, että vastaanottava palvelinsovellus tulkitsee niitä toisin kuin sovelluksen suunnittelija on tarkoittanut. Tyypillisesti syötteenä annettavaan merkkijonoon lisätään SQL-kielen erotinmerkkejä, joiden perässä on paljasta SQL-koodia, ja haavoittuva sovellus liittää syötteen osaksi tietokantakyselyään. Kuvat 1 ja 2 esittävät injektioiden periaatteen kuvitteellisella kannankäsittelykielellä.

RandomStormin Damn Vulnerable Web Application (DVWA) on helppo väline injektiohyökkäysten turvalliseen kokeiluun. DVWA esittää tyypillistä PHP:n ja MySQL:n päälle rakennettua verkkosovellusta, ja siinä esiintyvät haavoittuvuudet on jaettu useisiin "vaikeustasoihin". Artikkelissa perehdymme pääasiallisesti muutamiin konkreettisiin injektioesimerkkeihin sekä DVWA:sta löytyviin suojauksiin ja niiden sudenkuoppiin. Hyökkäysten teknisiä yksityiskohtia emme käsittele kuin pintapuolisesti. Artikkelin ja DVWA:n avulla aloittelijankin on helppoa kokeilla yksinkertaisia hyökkäystekniikoita, vaikka ei tuntisikaan PHP:tä ja SQL:ää. DVWA:n lukuisat muut mallihaavoittuvuudet tarjoavat pidempikestoista puuhaa edistyneemmillekin.

DVWA:n asennus

Skrolli on räätälöinyt Debian-pohjaisen käyttöjärjestelmän, jolle DVWA on asennettu valmiiksi. CD-ROM-levykuva on ladattavissa osoitteesta http://www.skrolli.fi/dvwa/. Levykuvan voi polttaa tyhjälle aihiolle esimerkiksi CD Burner XP:n avulla tai vaihtoehtoisesti tiedoston voi suoraan käynnistää virtuaalikoneeksi vaikkapa VMWare Playerin tai VirtualBoxin sisälle.

Levy sisältää kevyen työpöytäympäristön lisäksi Iceweasel-selaimen, yksinkertaisen web-palvelinympäristön ja apuskriptit DVWA:n käynnistämiseen. Alkuun päästäkseen käyttäjän tulee ensin tuplaklikata työpöydältä löytyvää "start_dvwa.sh"-ikonia, minkä jälkeen Iceweaselin käynnistäminen näyttää lisäohjeita tarjoavan aloitussivu.

DVWA:n injektiohaavoittuvuuksien lähdekoodit on esimerkkijakelulta löydettävissä hakemistosta"/opt/lampp/htdocs/dvwa/vulnerabilities/sqli/source/".

Yksinkertainen injektio

Kun DVWA:n turva-asetukset on säädetty alimmalle tasolle, voimme kokeilla yksinkertaista injektiota. DVWA:n sivulta ”SQL-injection” löytyy esimerkkisovellus, joka palauttaa halutun käyttäjän etu- ja sukunimen tietokannasta löytyvää käyttäjä-id:tä vastaan. Kuvassa 3 esitetään, mitä ohjelma palauttaa, kun sille annetaan mallisyöte.

Sovelluksen konepellin alla syöte liitetään sellaisenaan osaksi merkkijonoa, joka annetaan tietokannalle kyselynä. DVWA:n lähdekoodista nähdään, kuinka käyttäjän antama merkkijono muuttujassa "$id" upotetaan heittomerkeillä ympäröidyn SQL-merkkijonoliteraalin sisään. `Se puolestaan on osa SQL-kyselyä, jota rakennetaan lainausmerkeillä ympäröidyssä PHP:n merkkijonoliteraalissa:

```
$getid = "SELECT first_name, last_name
FROM users WHERE user_id = '$id'";
```

Kyselyn on tarkoitus kertoa, kenellä on käyttäjätunnus "$id". Koodista kuitenkin nähdään, että upotus on vaarallinen, jos "$id":n sisältämää merkkijonoa ei ole ennalta käsitelty turvalliseksi. Tätä voi testata antamalla User ID -kenttään tunnustelusyötteen:

```
' OR 'a'='a
```

Tunnustelusyöte aloitetaan heittomerkillä, joka sulkee SQL-kyselyn merkkijonoliteraalin. Loppuosa syötteestä on itse injektio, joka muokkaa kyselyn rakennetta. Kyselyn hakuehdoksi muodostuu:

```
user_id = '' OR 'a' = 'a'
```

Tämä ehto pitää paikkansa tietokantataulun jokaisen alkion kohdalla. Jos injektio onnistuu, palauttaa kysely kaikkien käyttäjien nimet. Tämä paljastaa, että kohdejärjestelmä on haavoittuva.

SQL-kyselykieltä tunteva voi käyttää näin löydettyä reikää tiedon keräämiseen kohdetietokannasta tai jopa ohjelmien ajamiseen heikosti suojatulla kohdekoneella.

Erikoismerkkien poisto

On ilmeistä, että äskeisen hyökkäyksen onnistuminen vaati, että syötteeseen saatiin heittomerkki, joka katkaisi SQL-merkkijonoliteraalin ennen injektio-osuutta. Tällaisia hyökkäyksiä on yksinkertaista torjua muuntamalla vaaralliset merkit vaarattomiksi ennen syötteen käsittelyä.

Kun Damn Vulnerable Web Application suojaukset säätää keskitasolle, ohjelma käsittelee syötemerkkijonon PHP:n "mysql_real_escape_string()"-funktiolla.

Jos ohjelmalle antaa nyt edellisen tunnustelusyötteen "' OR 'a'='a", DVWA esittää virheilmoituksen virheellisestä SQL-syntaksista:

```
"You have an error in your SQL syntax
... near '\' OR \'a\'=\'a\'' at line 1"
```

Uudesta virheilmoituksesta näemme, että "mysql_real_escape_string()" on lisännyt jokaisen heittomerkin eteen kenoviivan. Näin ne eivät lopeta SQL:n merkkijonoliteraalia. Mikäli käytetyt käsittelyfunktiot pystyvät luotettavasti karsimaan kaikki kyselyä mahdollisesti muokkaavat merkkijonot, siistimisfunktiot saattavat joissain tapauksissa olla riittävä tapa turvata syöte.

DVWA:n keskimmäisellä vaikeustasolla SQL-kysely on kuitenkin virheellinen, sillä "$id"-muuttujan arvoa ei upotetakaan heittomerkkien ympäröimään SQL-merkkijonoliteraaliin, vaan se liitetään paljaaltaan kyselyyn:

```
$getid = "SELECT first_name, last_name
FROM users WHERE user_id = $id";
```

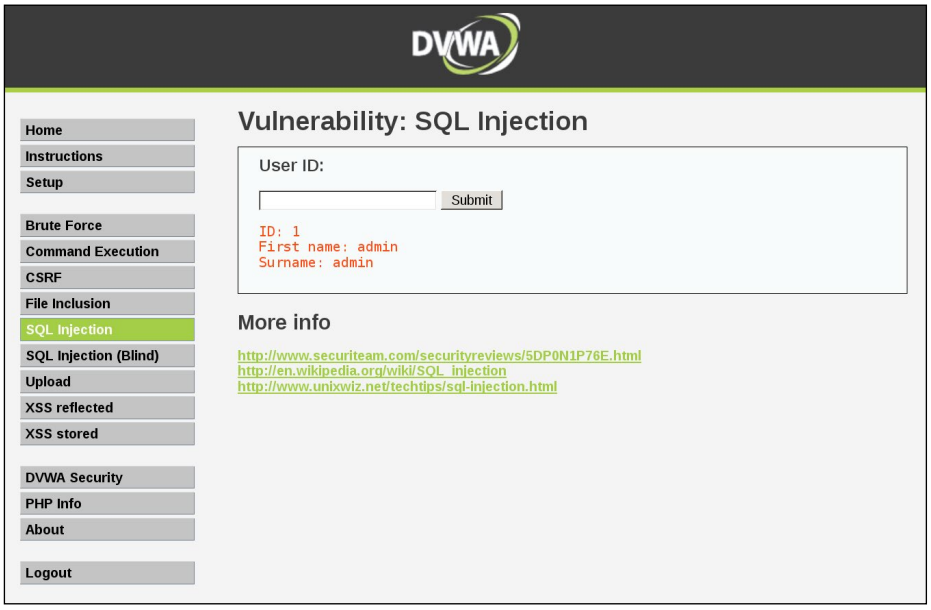
Tämän vuoksi DVWA antaa tunnustelusyötteellä virheilmoituksen eikä vain etsi käyttäjää, jolla olisi heittomerkkejä sisältävä tunnus.

Koska syötettä ei ole eristetty literaalin sisään, sovellukseen voi nyt hyökätä yksinkertaisella "1 OR 1 = 1" -injektiolla. Erikoismerkkien suojaaminen tekee kuitenkin monimutkaisisten injektioiden muotoilusta paljon vaikeampaa.

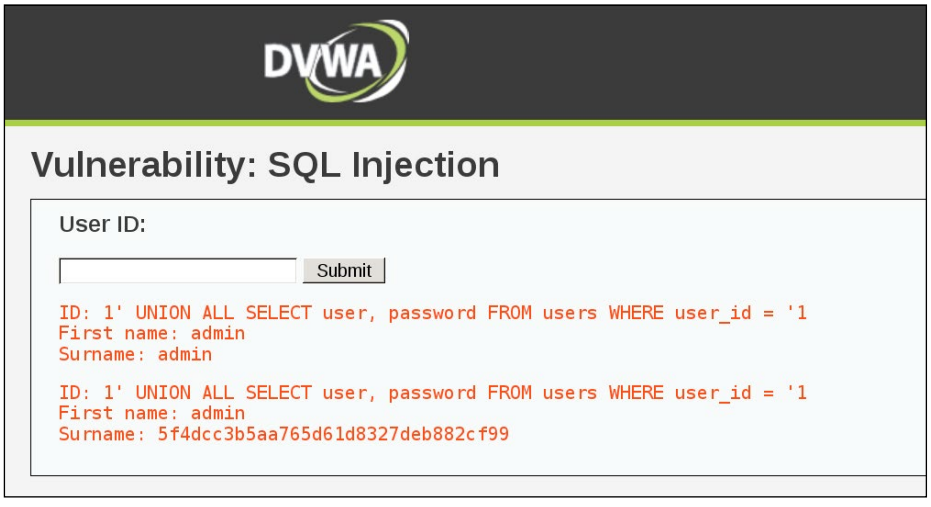
Paranoia+1

Vaikeimmalla tasolla DVWA käy nirsoksi. Ennen erikoismerkkien suojaamista ohjelma karsii syötteen kenoviivat ja varmistaa, että annettu syöte on numeerinen. Jos PHP:n tarkastusfunktiot toimivat, nämä keinot rajoittavat jo erittäin hyvin vahinkoa, jota epäilyttävät syötteiden aiheuttamaa vahinkoa.

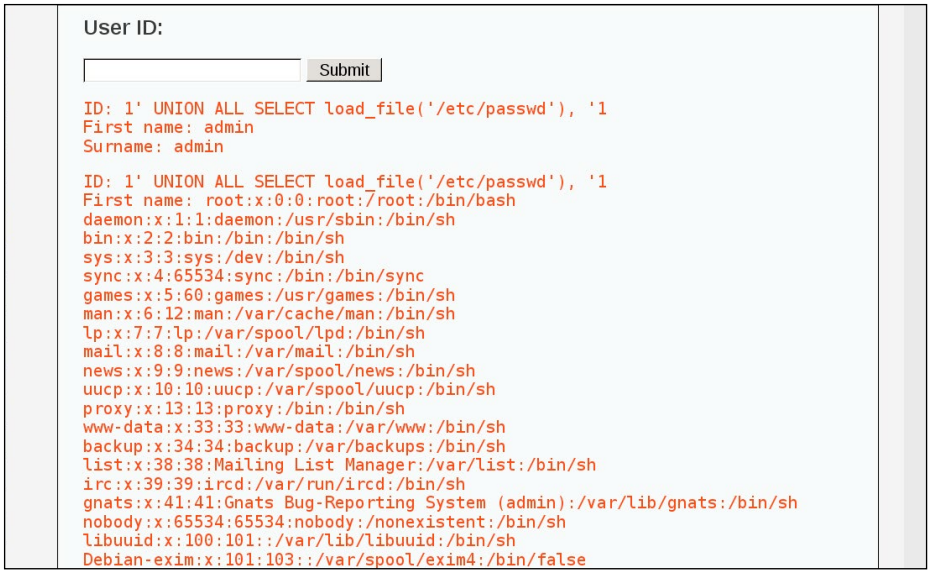
Käytännössä näin tarkka karsiminen on harvoin tarkoituksenmukaista, eivätkä kaikki tällaiset toimet tee DVWA:n suojauksesta turvallisempaa. Liian vainoharhainen suhtautuminen voi olla myös



Kuva 3: Miten malliohjelman 'tulisi' toimia.



Kuva 4: Syötteellä "1' UNION ALL SELECT user, password FROM users WHERE user_id = '1" voi tietokannasta kaapata pääkäyttäjän salasanatiivisteen.



Kuva 5: Heikosti suojatun tietokannan kautta pääsee käsiksi jopa järjestelmätiedostoihin.

haitallista, sillä useimpiin verkkopalvelujen syötekenttiin pitää pystyä syöttämään muun muassa pitkiä blogitekstejä, kommentteja ja koodinpätkiä. Esimerkiksi tämän artikkelitekstin tallentaminen tietokantaan verkkojulkaisua varten on mahdotonta, jos syötteestä hävitetään kirjaimet, lainausmerkit ja kenoviivat. Jos syötteet tarkastetaan pikkutarkasti aina ennen tietokantaoperaatioita, ohjelmoinnista tulee vaivalloista ja laskennasta raskasta.

Kyselyiden suojaus oikein

Oikea tapa estää injektiot on turvata suoritettava SQL-kyselyyn. MySQL, PostgreSQL ja muut käytetyimmät tietokantasovellukset tukevat esivalmisteltuja lauseita (*prepared statements*), joissa kyselyn muuttuvat osat pidetään erillään ympäröivästä kyselyrakenteesta. DVWA:n eri suojaustasojen haavoittuvuudet voi yksinkertaisemmin paikata esivalmistellulla kyselyllä näin:

```
# yhteys tulee ottaa laajennetun
mysqli-rajapinnan kautta
$connection = new mysqli("host",
"user", "pass", "dbname");
# määrittelee kyselyn muodon
$query = "SELECT first_name, last_name
FROM users WHERE user_id = ?";
# luo esivalmistellun kyselyn
$result = $connection->prepare($query);
# sitoo $id-syötteen kyselyyn integer-
tyyppisenä (i)
$result->bind_param('i', $id);
# suorittaa kyselyn ja tallentaa arvon
$result-olioon
$result->execute();
```

Tällöin tietokannalle annettu syöte "OR 'a' = 'a'" ei voi muuttaa itse kyselyn rakennetta. Koska se ei ole kokonaislukutyyppinen, syötteen sitominen kyselyyn epäonnistuu.

Myös tietokanta-abstraktioita tarjoavat välikerrokset ja sovelluskehykset käyttävät esivalmisteltuja lauseita ja muita sisällön säilyttäviä suodatusmenetelmiä. Esimerkiksi Djangossa käytetään oletusarvoisesti oliomaista relaatiomallinnusta (*object-relational* mapping, ORM) ja Ruby on Rails tarjoaa oman ActiveRecord-abstraktionsa. Jos tällaiset tietokanta-abstraktiot ovat riittävän tehokkaita, tarjoavat ne yleensä koetellun suojan injektioita vastaan. Muutoin on lähes aina syytä käyttää esivalmisteltuja lauseita.

Lähteet

XAMPP — Live-CD:llä käytetty web-palvelinohjelmisto
http://www.apachefriends.org/en/xampp.html
Damn Vulnerable Web Application
http://www.dvwa.co.uk/
HackYeah DVWA SQL Injection Walkthrough — artikkeli pohjautuu osin tähän
http://www.hackyeah.com/2010/05/hack-yeah-sql-injection-walkthrough-dvwa/
Open Web Application Security Project
https://www.owasp.org/
Bobby Tables — ohjeita SQL-injektioiden estämiseen eri ohjelmointikielillä ja sovelluskehyksillä
http://www.bobby-tables.com/

Kuivilla kääpistä, nyt

Dwarf Fortress on minulle maailman paras ja raivostuttavin peli. Se on suurin aikarosvoni irkin lisäksi. Seuraa rehellinen tilitys ongelmani syvyydestä.

Tapio Berschewsky

Eräs entinen kollegani sanoi minulle joskus itseni kannalta myrskyisenä vuonna 2007, että saattaisin tykätä Dwarf Fortressista. Mies tiesi, että olin uhrannut useita tuhansia tunteja elämästäni Nethackille melkein 20 vuoden saatossa.

En tarttunut, oli liian kiire muiden töiden lomassa pelata arvostelupelejä Xbox 360:llä. Muutaman päivän välein oli uusi peli, joka piti katsastaa lehteä varten. Se oli kyllä ihan hauskaa, mutta minkään kanssa ei päässyt yleensä perspuriloimaan kunnolla. Kääpiöt kytivät alitajunnassa kiinnostavana vaihtoehtona. Kunhan olisi aikaa.

Taisi olla joskus 2010 alkuvuodesta, kun toinen pitkäaikainen ystäväni sanoi, että ei ole varaa failata niin kuninkaallisesti, että jätän kääpiöt oikeasti katsomatta. Siirappia valuva ehdotus oli, että istuttaisiin yhdessä meillä iltaa. Sohva, projektori, Dwarf Fortress ja kaksi miestä. Opettaja ja oppilas tutustumassa ensiaskeliin.

Ensimmäinen peli

Maailma luotiin oletusasetuksilla, ja linnakkeen paikka valittiin lämpimältä alueelta joen viereltä. Ensimmäinen kääpiömäinen kokemus odotti heti nurkan takana. Ystävä neuvoi säätämään itse seitsemän siirtolaisen kyvyt ja alkutavarat paikalleen.

Kykypisteytys oli suhteellisen puuduttavaa, mutta ei vielä ahdistanut. Sitten vaihdettiin tavaroiden puolelle. Kaveri sanoi, että seuraavaksi valitaan yksi kappale jokaista tynnyrissä tulevaa ruokaa, joka maksaa kaksi pistettä. Niitä on monta ja valikko on hirveä.

Tyhjä tynnyri maksaa 10 pistettä, joten puolen tunnin vaivalla tuli varmistettua yhden illan linnakkeelle heti kättelyssä pitkiksi ajoiksi riittävä halpa tynnyrivarasto. Olin kokemuksen jälkeen hieman nöyrempi ja tallensin joukkioni asetukset tulevia pelejä varten.

Illan aikana kirosin käyttöliittymän alimpaan helvettiin moneen kertaan. Louhin ensimmäisen käytäväni, tein ensimmäisen farmini maan alle, ja aloitin makuuhuoneiden värkkäämisen. Seuraavana päivänä aloin uuden linnakkeen tyhjästä omin neuvoin. Se olikin sitten menoa.

Vuosi addiktiota

En ole antanut millekään pelille niin paljon aikaa vuosiin kuin Dwarf Fortressille. Rakkaussuhde syntyi välittömästi. Nyt eri versioiden tallennuskansioissa on useita kymmeniä linnakkeita. Jokainen niistä on merkki uusien asioiden oppimisesta ja vanhojen paremmasta toteutuksesta.

Pelasin ensin 0.31.xx-versioita, jatkuvasti enemmän. Otin miniläppärille kääpiöt mukaan jopa häämatkalle Firenzeen, koska hei, lentokoneessa ehtisi ihan hyvin pelaamaan pientä yhden ruudun kokoista alle 20 kääpiön linnaketta. Pelasin sitten parina aamuyönäkin, mutta selvisin nuhteluitta.

Sen sijaan en selvinnyt kovin pitkiä aikoja ilman Döffiä. Aika harvassa oli viikko, jonka aikana ei syntynyt uutta linnaketta jolle antoi ainakin 10-15 tuntia aikaa. Kun oikein kunnolla pääsi makuuhuoneiden sisustamisen makuun, huomasi olevansa hereillä paitsi vielä aamulla kun vaimo lähti töihin, myös iltapäivästä vaimon tullessa kotiin. Ei nyt kovin usein, mutta silti.

Hetkellinen eheytyminen

Sitten hyytyi into. 0.31 ei enää päivittynyt, Toady oli siirtynyt työskentelemään suljettujen ovien takana. Uusi versio oli luvassa joskus, kun hillitön kasa uusia ominaisuuksia olisi lisätty peliin. Saman version jatkuva pelaaminen alkoi tuntua tylsältä.

On suhteettoman epäreilua lisätä peliin jatkuvasti uusia tajunnanräjäyttävältä kuulostavia ominaisuuksia, blogata niistä yksityiskohtaisesti, ja olla julkaisematta. Vaikka pelasinkin pääosin kaikenlaista muuta, oli kääpäkoukku niin syvällä, että luin parin päivän välein jokaisen päivityksen blogista. Irkkasin myös määrättömiä määriä kääpiöistä, auttaen aloittelijoita ja oppien guruilta.

Vuoden vaihtuessa toista kertaa Toady ihan oikeasti alkoi kuulostaa siltä, että julkaisu tulee pian. Kun 0.34.01 tuli ulos helmikuussa 2012, tiputin hanskat siihen paikkaan, ja pelasin pari vuorokautta putkeen. Linnakkeisiin ilmestyi verta imeviä vampyyreja! Kaikenlaista muutakin, mutta varsinkin vampyyreja!

Takaisin pohjalla

Kun hurahdin 0.34.xx-versioihin, aiempi Döffi-koukkuni alkoi näyttämään naurettavalta pikku harrastelulta. Nyt ei enää tarvinnut pelata töikseen lähes ollenkaan, ja vapaa-aikaa oli enemmän kuin aiempina vuosina.

Viikot vierivät parempia ja parempia kaivoksia, asuinalueita, portteja, parakkeja ja muita linnakkeen osia suunnitellessa. Päivityksiä tuli tasaiseen tahtiin, ja uudet tasapainotukset olivat miellyttäviä.

Linnoituksista alkoi tulla kokoajan yksityiskohtaisempia, automatisoidumpia ja uskaliaampia. Aloin voittamaan isoja sotia ja megapetoja koulutetulla armeijallani. Pian kaikki vastus katosi pelistä. Jokainen uusi uljas linnake kuoli lopulta tarkoituksselliseen itsemurhaan, kätketyn kuoleman kammion avaamiseen.

Syksyllä päivitykset ehtyivät. Jaksoin pari kuukautta, mutta sitten kääpiö ei enää innostanut. Oli mielenkiintoisempi tehdä Skrollia kuin pelata vanhaa Döffiä. Olen sittemmin yrittänyt eheytyä muun muassa antamalla kaiken Döffiltä vapautuneen aikani Minecraftille, ja olemalla suunnittelematta enää parempia linnoitusmalleja pääni sisällä.

Katsotaan onnistuisiko seuraavan julkaisun kohdalla pitämään pään kylmänä. Luen kyllä yhäkin blogia.

Ei näin!

Konix Multi-System, vallankumous on peruttu

Jo 80-luvulla bittiä nyplänneet voivat muistaa peliohjainvalmistaja Konixin. Erityisesti Speed King -ohjain oli täkäläisittäinkin tunnettu. Konix kuitenkin tähtäsi myöhemmin tähtiin, epäonnistuen julmasti. Teksti: Mikko Heinonen Kuvat: Paul Neal ja Steve Gallichan

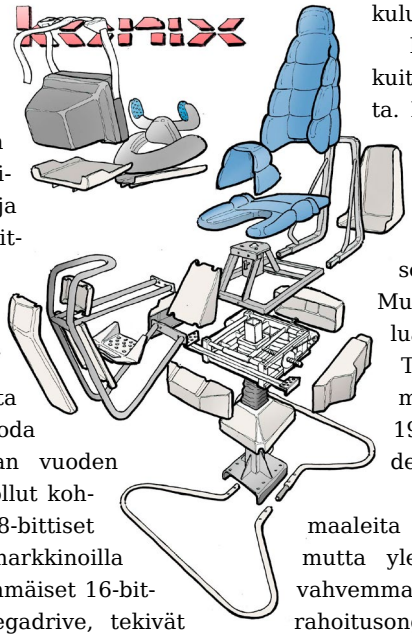
Vuosikymmenen lähestyessä loppuaan Konixilla oli suuria suunnitelmia. Pelkkien tikkujen teon sijaan se halusi luoda aivan uudenlaisen, Slipstream-nimisen lisälaitteen. Slipstreamin oli tarkoitus olla muunneltava ohjain, josta saisi vaikkapa auton ratin tai lentokoneen sauvaohjaimen tarpeen mukaan. Tarjolla oli jopa kuluttajatuotteissa ennennäkemätön force feedback.

Melko pian heräsi ajatus siitä, että Slipstreamista saisi aikaan kokonaisen pelikonsolin. Pelkkään pelaamiseen tarkoitettujen koneiden myynti oli lähtenyt Euroopassakin uudelleen nousuun lähinnä Nintendon vetämänä, ja osille halusi jokainen kynnelle kykenevä.

Samaan aikaan Sinclair Researchin leivistä oli irtautunut Flare Technology -niminen porukka, joka jatkoi Sinclairin Loki-projektia. Lokista oli pitänyt tulla Amigan ja Atari ST:n kanssa tasaveroisesti kilpaillut supermikro, joka olisi Amigan tapaan perustunut varsinaista suoritinta tukeviin erikoispiireihin. Nyttemmin Flare Oneksi ristityn koneen ytimessä sykki vanhahtava Z80, mutta räätälöidyn grafiikkaraudan ansiosta se pärjäsi vaikkapa 3D-laskennassa jopa ARM-pohjaiselle Acorn Archimedekselle.

Kahdesta heikosta yksi

Flare Technologylla ei käytännössä ollut resursseja tuoda Flare Onea tuotantoon, joten heille sopi varmasti hyvin, kun Konixin johto lähestyi suunnittelijoita yhteistyömielessä. Flare One oli juuri jotain sellaista, mitä Slipstreamin sisään oli kaavailtu. Vanhan sotaratsun Z80:n tilalle Konix tosin tahtoi modernimman 8086:n, jotta konsolista saatiin muodikkaasti 16-bittinen. Multi-Systemiksi nimetyn koneen kehitys alkoi toden teolla vuonna 1988, ja hyvin pian lehtiinkin alkoi tipahdella ennakkotietoja mullistavasta pelilaitteesta.



Kun Multi-Systemin ensimmäinen versio esiteltiin messuilla alkuvuodesta 1989, tarkoitus oli tuoda se markkinoille saman vuoden jouluksi. Ajoitus olisi ollut kohtalaisen hyvä, sillä 8-bittiset konsolit olivat olleet markkinoilla jo useita vuosia. Ensimmäiset 16-bittiset, kuten Sega Megadrive, tekivät vasta tuloaan. Myös hinnan piti jäädä 200 punnan tietämiin, mitä saattoi pitää hyvin kohtuullisena.

Eräänä kuumottavana yksityiskohtana laitteen yhteydessä oli tarkoitus esitellä myös siihen sopiva pelituoli, joka liikkuisi pelitapahtumien mukaan. Tällaista oli nähty vain hulvattoman kalliissa peliautomaateissa, mutta Konixin vakaa aikomus oli tuoda tuolikin markkinoille kuluttajahintaluokassa.

Kahdesta pienestä ei kuitenkaan tullut yhtä suurta. Konixilla oli kokemusta peliohjaimista, mutta pelikonsolin tuominen markkinoille on kertaluokkaa monimutkaisempi operaatio. Niinpä Multi-Systemin suunnitteluaikataulu alkoi lipsua. Tavoite joulusta 1989 muuttui pian keväälksi 1990, sitten saman vuoden syksyksi.

Viivästykset ovat normaaleita tuotekehityksessä, mutta yleensä sitä tehdäänkin vahvemmalla kassalla. Konixin rahoitusongelmat etenivät nopeas-

ti siihen pisteeseen, että se joutui myymään tärkeimmän pääomansa, peliohjaintensa valmistusoikeudet, saadakseen rahaa Multi-Systemin tekemiseen.

Moninkertaiset pulmat

Pelikonsoliksi poikkeuksellista oli, että Multi-System suunniteltiin käyttämään levykkeitä ROM-moduulien sijaan. Tämä oli julkaisijoille helpotus, koska pelin kopiointi levykkeelle maksoi murto-osan ROM-piirien hankinnasta ja moduulien prässäämisestä. Samalla se kuitenkin aiheutti ongelman, sillä pelit piti ladata levykkeeltä koneessa olevaan muistiin.

Koska muistipiirit olivat kalliita, prototyypissä oli RAMia vain 128 kilotavua — neljäsosa Amigan ja Atari ST:n perusversiosta. Kehittäjät moittivat ratkaisua, koska se käytännössä edellytti, että kone lataa levykkeeltä jatkuvasti. Hätäkorjauksena Konix ilmoitti tuovansa markkinoille laajennuksen heti, kun muistipiirien hinta vain laskisi.

Todellisuus oli muutenkin haavekuvia karumpaa. Moottoroitu tuoli saatiin kyllä aikaan, mutta yleisölle esitelty prototyyppi oli tehty puusta, auton istuimesta ja porakoneista. Lähes kaikki laitteelle ilmoitetut pelit olivat käännöksiä Amigalta tai Atari ST:ltä ja vain harva niistä hyödynsi Multi-Systemin erikoisia ohjauslaitteita. Paras pelattavaksi asti päässyt peli oli ilmeisesti Jeff Minterin Attack of the Mutant Camels. Kehitystyötä vaikeutti entisestään se, etteivät Konix ja Flare saaneet aikaan lopullista tuotantomallia laitteesta.

Lopullisesti Multi-Systemin kuitenkin kaatoi Konixin rahoittajien huumorintajun loppuminen. Rahaa laitteen tuotantoversion tekemiseen ei kertakaikkiaan enää ollut, joten odotettua konetta ei koskaan nähty kauppojen hyllyillä. Huhutaan, että osansa asiaan oli myös laitteen suunnittelijoiden ylpeydellä, sillä Konix halusi loppuun asti tehdä laitteen yksin, eikä myydä sen oikeuksia kenellekään.

Pesänjakajat

Multi-System olisi voinut olla aidosti eurooppalainen superkonsoli, mutta todellisuus juoksi sen kiinni ennen loppusuoraa. Yhtään lopullista laitetta ei ole tallessa, koska sellaista ei koskaan saatu kasaankaan. Emulaattori ja sekalainen kokoelma koneen demoamiseen käytettyjä ROM-tiedostoja verkosta kyllä löytyy, mikä sinällään on upeaa ja osoitus harrastajayhteisön voimasta. Videopalveluista löytyy demojen lisäksi pätkiä muun muassa moottorituolin rakentamisesta.

Slipstream-ohjain, tosin selvästi yksinkertaistettuna, sen sijaan jopa oikeasti julkaistiin. Kiinalaisten valmistaman ohjaimen helposti muistettava lopullinen nimi oli Multi-System Super MS-200E. Näihin ovat harrastajat törmänneet ainakin Tanskassa.

Flare Technology jatkoi toimintaansa Konixin kaatumisen jälkeenkin, ja seuraavaksi sen palveluksia kaivattiin Atlantin toisella puolen. Entinen suuruus Atari nimittäin tarvitsi uuden kotikonsolin — jonka tarina kerrottiin tällä palstalla numerossa 2013.1.

Koska Konix Multi-System ei koskaan valmistunut, siitä on olemassa vain prototyypin kuvia. Artikkelin kuvat antoivat ystävällisesti käyttöömme konsolin muotoilusta vastanneen Level Six Design Consultantsin Paul Neal ja Steve Gallichan, jotka toimivat nykyisin Product Partners -yrityksen johtajina. Product Partnersin sivuilla on lisääkin konseptikuvia Konixista: www.productpartners.co.uk

OpenGL:n perusteet

Viime numerossa luotiin lyhyt katsaus 3D-grafiikkarajapintojen historiaan. Nyt menemme itse asiaan ja tutustumme OpenGL:n käyttöön.

Teksti ja kuvat: Mikko Rasa

Miksi OpenGL?

3D-ohjelmointia aloittelevalle ensimmäinen valinta on käytettävä rajapinta. Mikäli alkeita ei haluta opetella aivan pikselitasolta alkaen, on vaihtoehtoja nykypäivänä käytännössä kaksi: OpenGL ja Direct3D. Myös rautakiihdytykseen pääsee nykykoneilla käsiksi vain valmiiden rajapintojen kautta.

OpenGL on nimensä mukaisesti avoin standardi, jonka kehitystä ohjaa piiri- ja ohjelmistovalmistajista muodostettu komitea, ARB — Architecture Review Board. Siitä on olemassa useita toteutuksia monille eri alustoille, mukaan lukien avoimen lähdekoodin Mesa. Mobiililaitteilla käytetty OpenGL ES on oma standardinsa mutta läheistä sukua OpenGL:lle. Eroja on lähinnä yksityiskohdissa ja tuetuissa ominaisuuksissa. Pienellä etukäteissuunnittelulla on mahdollista kirjoittaa koodia, joka toimii lähes kaikilla alustoilla.

Direct3D puolestaan on Microsoftin kehittämä rajapinta. Se lanseerattiin alun perin Windows 95:n mukana osana DirectX-rajapintaperhettä. Pakettiin kuuluu 3D-grafiikan lisäksi myös muita erityisesti peliohjelmoinnissa tarvittavia rajapintoja. Windows Phonella ja Xboxilla Direct3D on ainoa tarjolla oleva 3D-grafiikkarajapinta.

Suorituskyvyn ja ominaisuuksien osalta rajapinnat ovat tasaväkiset. Eri näytönohjainvalmistajien toteutusten välillä tosin on eroja. Nvidian OpenGL-tuki on perinteisesti ollut kattavin ja toimivin, mutta ajureissa on joitakin outouksia. ATI:n ostaneen AMD:n toteutuksissa puolestaan törmää silloin tällöin suoranaisiin bugeihin. Nvidian ohjaimet ovat näin ollen parempia kehitysvaiheessa, mutta tuotokset on parasta testata muillakin alustoilla ennen levitystä.

Miten OpenGL toimii

OpenGL:ssä kappaleet koostetaan tasokuvioista, jotka renderöidään pikseleiksi tekniikalla, jota kutsutaan rasteroinniksi. Rasterointi on useimmiten hyvin taloudellinen renderöintitapa verrattuna ray tracing -tekniikkaan. Haittapuolena kuitenkin on, että aidosti pyöreitä kappaleita on mahdoton tehdä, ja monien valotehosteiden toteuttaminen on monimutkaista.

OpenGL on tilakone. Siinä on suuri määrä erilaisia tilamuuttujia, kuten käytettävät shader-ohjelmat, tekstuurit ja kappaleen datan sijainti muistissa. Tarvittavat muuttujat on asetettava ennen piirtokomennon suorittamista. Tilat kuitenkin pysyvät muuttumattomina, kunnes ne asetetaan uudestaan, joten useita samanlaisia kappaleita voidaan helposti piirtää useita peräkkäin. Samoilla tiloilla ja samalla datalla suoritettava piirtokomento tuottaa aina saman lopputuloksen.

Kolmiulotteisen kappaleen muodostamiseksi tarvitaan kahdentyyppistä dataa: kulmapisteet eli verteksit ja primitiivit eli alkeiskuviot. Kunkin kulmapisteen attribuutit määrittelevät sen sijainnin, värin, tekstuurikoordinaatit ja muut ominaisuudet. Primitiivit puolestaan kertovat, kuinka kulmapisteiden väliin muodostetaan renderöitäviä pintoja. Primitiivit

> Ennen versiota 3.0 OpenGL tarjosi joitakin valmiiksi määriteltyjä attribuutteja. Näistä kuitenkin luovuttiin rajapinnan yksinkertaistamiseksi. Modernissa OpenGL-koodissa on ohjelmoijan tehtävä tarjota tulkinta attribuutille kullekin.

määritellään viittaamalla kulmapisteisiin indekseillä (engl. index), jotka toimivat kuten taulukon indeksit. Omia ominaisuuksia ei primitiiveillä ole.

OpenGL:ssä kaikki pinnat määritellään kolmioina, koska se on matemaattisesti yksinkertaisin käyttökelpoinen muoto ja helpottaa siten algoritmien suunnittelua ja toteutusta. Monimutkaisemmat tasopinnat on pilkottava kolmioiksi renderöintiä varten. Myös likimain pyöreitä pintoja voidaan muodostaa monista pienistä kolmioista. Oikein tehtynä eroa aidosti pyöreään pintaan ei juuri huomaa.

Kolmiot voivat olla irrallisia tai osana kolmioketjua (engl. triangle strip) tai viuhkaa (engl. triangle fan). Niissä olevat kolmiot jakavat kulmapisteitä keskenään, joten saman kolmiomäärän piirtämiseen tarvitaan pienempi määrä indeksejä. Suurissa kappaleissa tämä voi vähentää merkittävästi indeksitaulukoiden kokoa ja siten parantaa suorituskykyä. Usean erillisen ketjun tai viuhkan piirtäminen yhdellä kutsulla vaatii kuitenkin pientä kikkailua.

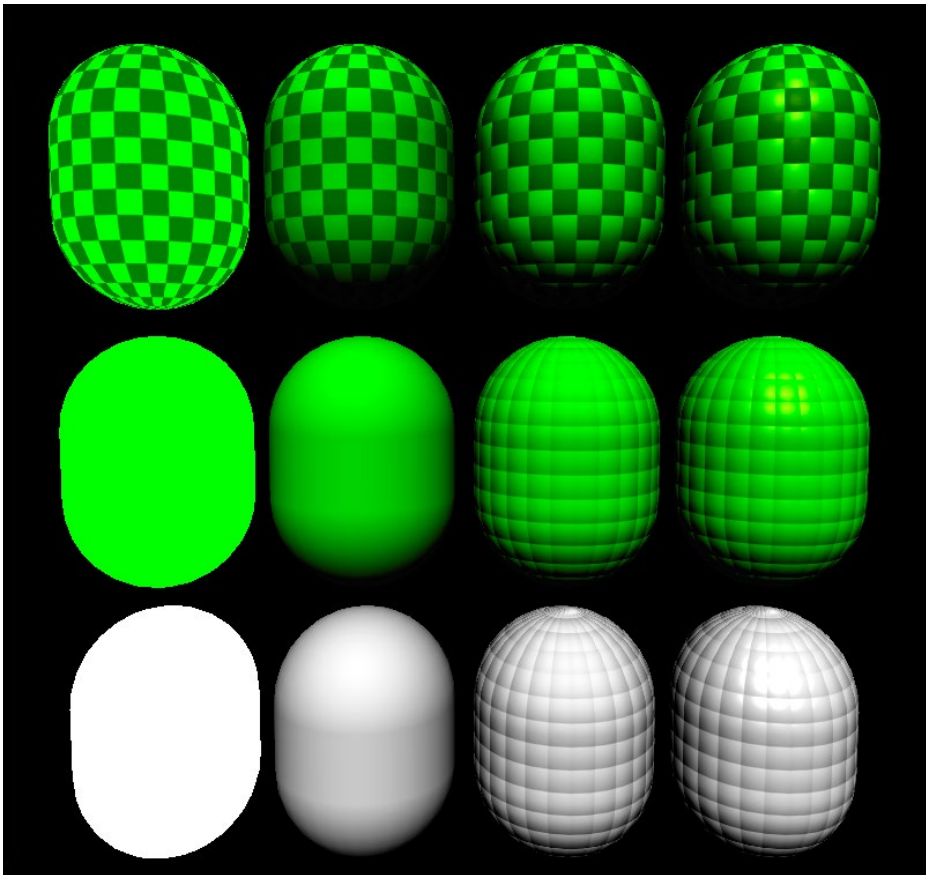
Kolmioiden lisäksi on mahdollista piirtää myös viivoja ja pisteitä. Viivat voivat kolmioiden tapaan olla irrallisia tai osana ketjua (line strip) tai silmukkaa (line loop). Pisteet ovat aina irrallisia. Nämä primitiivit ovat harvinaisempia kuin kolmiot, mutta pisteitä käytetään usein ns. partikkeliefektien piirtämiseen.

Kappaleen data on vielä muutettava kuvaruudun koordinaateiksi ja väreiksi. Tästä huolehtivat shaderit. Ne ovat lyhyitä ohjelmia, joita suoritetaan näytönohjainpiirillä. Shaderit kirjoitetaan C:tä muistuttavalla GLSL-ohjelmointikielellä (GL Shading Language).

Shader-ohjelmien suorituksessa on muutamia eri vaiheita. Ensimmäisenä suoritetaan kulmapisteshaderi (vertex shader), jonka tärkein tehtävä on muuntaa kulmapisteiden sijainnit ruudun koordinaatistoon. Se myös valmistelee verteksien muut ominaisuudet myöhempiä vaiheita varten. Raskaat laskentaoperaatiot kannattaa sijoittaa kulmapisteshaderiin, koska se suoritetaan huomattavasti harvemmin kuin pikselishaderi.

Seuraavana vuorossa on geometriashaderi (geometry shader). Se käsittelee kokonaisia primitiivejä, ja voi muuttaa niiden tyyppiä tai määrää. Oikein käytettynä geometriashaderilla voi vähentää pääsuorittimen ja väylän kuormitusta huomattavasti tietynlaisissa tehosteissa. Tämä vaihe ei ole pakollinen, ja perehdymme siihen tarkemmin myöhemmässä artikkelissa.

Viimeinen vaihe on pikselishaderi (fragment shader tai pixel shader), joka suoritetaan kerran jokaiselle rasteroitavan pinnan pikselille. Sen syötteitä interpoloidaan pinnan yli, mikä mahdollistaa pehmeät väriliu'ut, jatkuvan teksturoinnin ja monenlaiset pikselikohtaiset valotehosteet. Myös laskennallista grafiikkaa voidaan luoda pikselishadereilla. Pikselishadereista on enemmän tietoa toisaalla tässä numerossa.



Eri shader-malleja

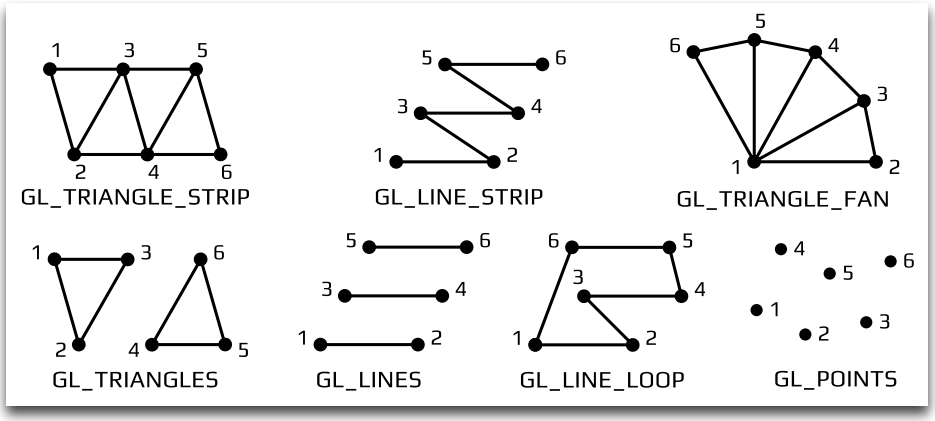
tyyppi	pääte	vakio
char	b	GL_BYTE
unsigned char	ub	GL_UNSIGNED_BYTE
short	s	GL_SHORT
unsigned short	us	GL_UNSIGNED_SHORT
int	i	GL_INT
unsigned int	ui	GL_UNSIGNED_INT
float	f	GL_UNSIGNED_INT
float	f	GL_FLOAT
double	d	GL_DOUBLE

Yleisimpiä tietotyyppejä. Mikäli funktion nimen lopussa on v, se ottaa pointterin kyseisen tyyppisen dataan.

OpenGL:n ohjelmointirajapinta

OpenGL-standardi on suunniteltu C-kieltä silmälläpitäen. Monille muillekin ohjelmointikielille on tehty rajapintasidoksia, jotka yleensä muistuttavat enemmän tai vähemmän C-kielistä rajapintaa. Tämän artikkelin esimerkit on tehty C++:lla, mutta soveltaminen muihin ohjelmointikieliin ei ole vaikeaa. Joissakin osissa toteutus voi olla hieman erilainen johtuen esimerkiksi osoittimien puutteesta.

Monet OpenGL:n funktiot ottavat pa-



OpenGL-primitiivejä

rametreja, joilla on ei-numeerinen merkitys. Tällaisia ovat esimerkiksi tekstuurin väriformaatti tai shaderin tyyppi. Näitä arvoja esitetään nimetyillä kokonaislukuvakioilla.

Jotta ohjelman antama data tulkittaisiin oikein, on myös sen tietotyyppi kerrottava OpenGL:lle. Tähän on kaksi eri tapaa. Joillakin funktioilla on nimen perässä pääte, joka kertoo minkä tyyppisiä arvoja kyseinen funktio ottaa. Näistä funktioista on yleensä useita eri muunnelmia eri tietotyypeille. Osa funktioista taas ottaa yhtenä parametrina tietotyyppiä kuvaavan lukuvakion.

OpenGL-standardi sallii toteutuksen ylläpitää useita virhekoodeja vähentääkseen synkronoinnin tarvetta hajautetussa renderöinnissä. Tällä on merkitystä lähinnä suurissa renderöintifarmeissa, ja käytännössä siihen ei kuluttajatason raudalla törmää.

OpenGL on monelta osin oliopohjainen rajapinta, mutta olioiden tietorakenteet on piilotettu ohjelmoijalta. Niihin viitataan käyttäen kokonaislukutunnisteita. Oliot on erikseen luotava ennen niiden käyttöä, ja ne on myös muistettava tuhota käytön jälkeen. C++:lla tämä onnistuu kätevimmin, kun jokaiselle oliotyypille tekee luokan, joka huolehtii alla olevan OpenGL-olion elinkaaresta.

Jotta oliota voisi käyttää piirtämiseen, se on sidottava (bind) aktiiviseksi. Monilla oliotyypeillä on useita sidontapisteitä (binding point), mahdollistaen esimerkiksi monen tekstuurin samanaikaisen käytön. Useimpia olioita ei myöskään voi käsitellä suoraan, vaan operaatiot kohdistuvat aktiiviseen olioon.

Mikäli OpenGL-funktiokutsussa tapahtuu virhe, ohjelman on erityisesti kehitysvaiheessa hyvä kyetä raportoimaan siitä. Tätä varten on funktio glGetError. Sen palauttama lukuvakio ilmaisee, millainen virhe edellisen kutsukerran jälkeen on tapahtunut. Jos virheitä tapahtuu glGetError-kutsujen välissä useampia, ainoastaan ensimmäinen virhe säilytetään. Myöskään virhekoodista ei selviä, mikä funktiokutsu sen aiheutti.

Yksinkertaisen OpenGL-ohjelman anatomia

Ennen kuin mitään muita OpenGL-kutsuja voi tehdä, ohjelman on luotava konteksti. Se on superolio, joka pitää sisällään OpenGL:n koko tilan. Ohjelmalla voi olla samanaikaisesti useita kontekseja, mutta kullakin säikeellä voi olla vain yksi aktiivinen konteksti. Sama konteksti voi vastaavasti olla aktiivisena vain yhdessä säikeessä kerrallaan.

Grafiikan saamiseksi ruudulle tarvitaan myös ikkuna. Tapa kontekstin ja ikkunan luomiseen riippuu järjestelmästä, ja sivuutamme yksityiskohdat tässä artikkelissa. Skrollin nettisivuilta löytyy esimerkkejä joillekin suosituimmille järjestelmille ja kirjastoille.

Jotta voisimme piirtää ikkunaan jotain, tarvitsemme aluksi kulmapisteet ja primitiivit. Helpoin tapa kulmapisteiden määrittelyyn on luoda tietorakenne, joka pitää sisällään yhden kulmapisteen ominaisuudet. Koska yhdellä kulmapisteellä ei tee mitään, määrittelemme niitä taulukollisen. Indeksejä emme tässä esimerkissä tarvitse, koska OpenGL tarjoaa erillisen funktion peräkkäisten kulmapisteiden piirtämiseen.

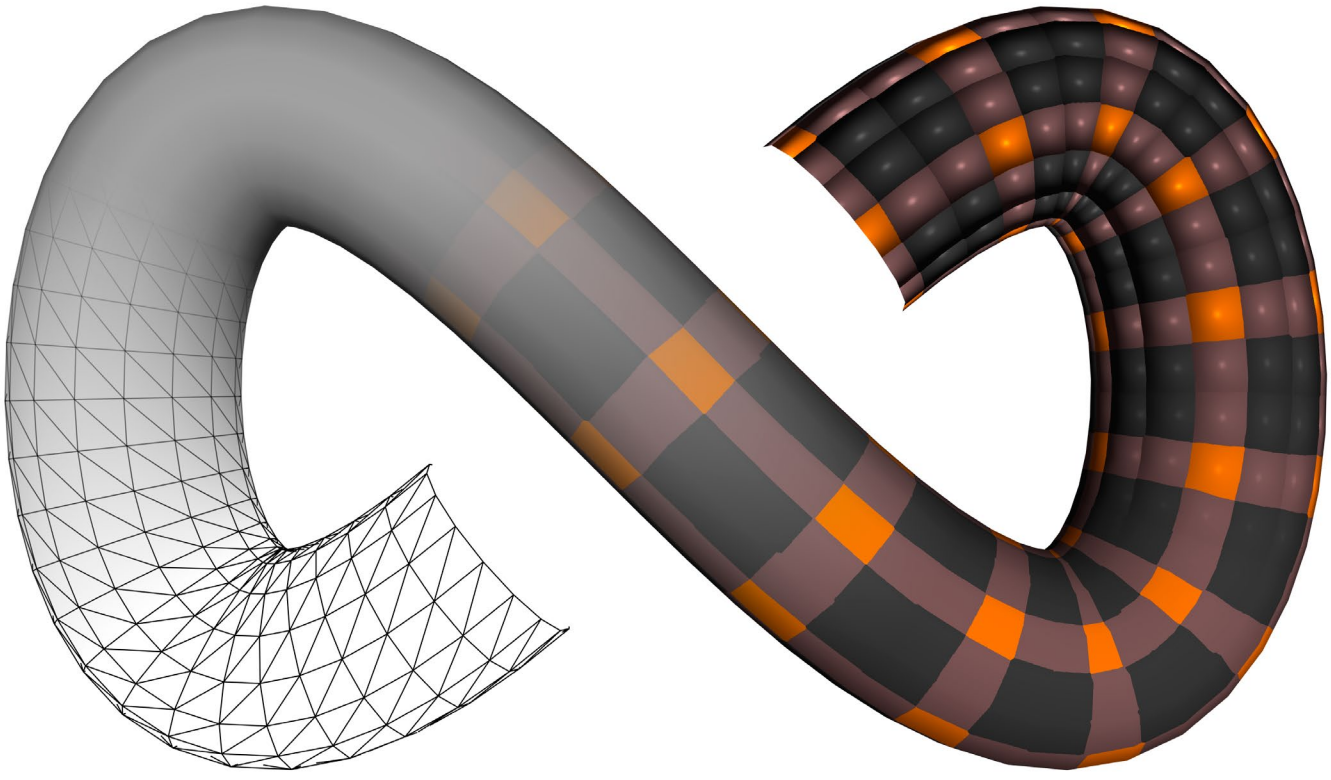
```
struct Vertex
{
  float x, y;
};

Vertex vertices[] =
{
  { -0.5,  0.5 },
  { -0.5, -0.5 },
  {  0.5,  0.5 },
  {  0.5, -0.5 }
};
```

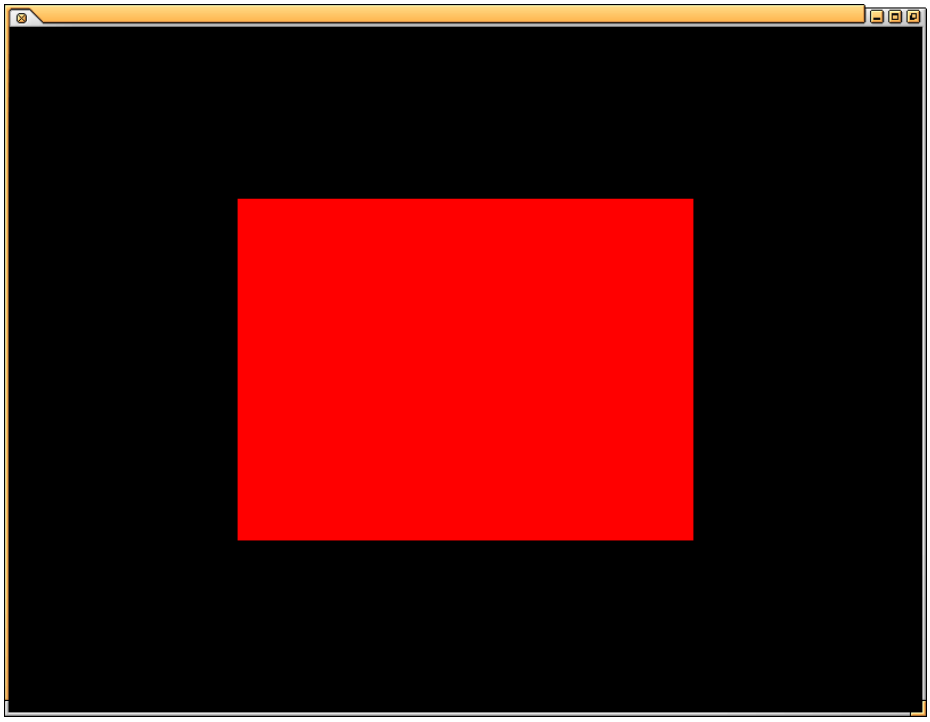
OpenGL:n oletuskoordinaatistossa ruudun koordinaatit ovat välillä [-1, 1], missä X kasvaa oikealle ja Y ylöspäin. Yllä olevat kulmapisteet määrittelevät näin ollen suorakulmion, joka on kummassakin suunnassa puolet ruudun koosta.

Seuraavaksi luomme puskurin ja lataamme kulmapisteet siihen:

```
unsigned vbuf_id;
glGenBuffers(1, &vbuf_id);
glBindBuffer(GL_ARRAY_BUFFER, vbuf_id);
glBufferData(GL_ARRAY_BUFFER,
             sizeof(vertices),
             vertices,
             GL_STATIC_DRAW);
```



Kolmiulotteinen kappale vaihe vaiheelta: kolmioverkko, väripinnat, tekstuurit, varjostus.



Kuvakaappaus esimerkkiohjelmasta

glBufferData-kutsun viimeinen parametri on vihje datan käyttötavasta. Tässä tapauksessa aiomme määritellä datan kerran ja sen jälkeen käyttää sitä monta kertaa piirtämiseen. Vihjeen perusteella ajurit voivat päättää, mille muistialueelle data sijoitetaan. Väärän vihjeen antaminen ei tee ohjelmasta toimimatonta, mutta saattaa hidastaa toimintaa.

Koska kulmapisteiden koordinaatit ovat jo valmiiksi oikeassa koordinaatistossa, tarvittavat shaderit ovat hyvin yksinkertaiset. Verteksivarjostin kopioi sille annetun koordinaatin OpenGL:n tarjoamaan muuttujaan. Pikselishaderi asettaa jokaisen pikselin väriksi kirkkaan punaisen.

```
in vec4 in_position;
void main()
{
  gl_Position = in_position;
}
```

Vertex shader

```
void main()
{
  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}
```

Fragment shader

vec4 on vektorityyppi, joka koostuu neljästä liukuluvusta. Aiemmin määrittelemissämme kulmapisteissä on vain kaksi koordinaattia, mutta tämä ei haittaa, koska OpenGL täyttää loput koordinaatit oletusarvoilla (0, 0, 0, 1). Neljättä koordinaattia käytetään perspektiivin muodostamiseen; perehdymme taustalla olevaan matematiikkaan seuraavassa osassa.

Shadereita ei voi suorittaa lähdekoodimuodossa, vaan ne on ensin käännettävä. OpenGL sisältää GLSL-kääntäjän, jota käytetään funktiokutsujen kautta. Useimmista muista OpenGL:n olioista poiketen varjostimia voi käsitellä suoraan sitomatta niitä aktiiviseksi.

```
unsigned
vs_id = glCreateShader(GL_VERTEX_
SHADER);
glShaderSource(vs_id,
  // Lähdekoodi koostuu
  // yhdestä merkkijonosta
  1,
  // Merkkijonojen osoitteet
  &vertex_shader_src,
  // Merkkijonojen pituudet;
  // ei tarvita, jos merkkijonot
  // ovat terminoituja
  NULL);
glCompileShader(vs_id);
```

Käännöksen onnistuminen on hyvä tarkistaa ja mahdolliset virheet tulostaa:

```
int status;
glGetShaderiv(vs_id, GL_COMPILE_STATUS,
              &status);
if(!status)
{
  char buf[1024];
  glGetShaderInfoLog(vs_id,
    // Annetun puskurin koko
    sizeof(buf),
    // Palautusmuuttuja merkkijonon
    // pituudelle; palautettava
    // merkkijono on terminoitu,
    // joten tätä ei tarvita
    NULL,
    buf);
  printf(
"Failed to compile shader:\n%s\n",
    buf);
}
```

Pikselishaderi käännetään vastaavasti, mutta glCreateShader-kutsulle annetaan parametrina vakio GL_FRAGMENT_SHADER. Kulmapiste- ja pikselishaderit on vielä liitettävä yhteen shaderohjelmaksi:

```
enum
{
  POSITION = 0
};

unsigned prog_id = glCreateProgram();
glAttachShader(prog_id, vs_id);
glAttachShader(prog_id, fs_id);
glBindAttribLocation(prog_id, POSITION,
  "in_position");
glLinkProgram(prog_id);
```

glBindAttribLocation-kutsu sitoo kulmapisteshaderille määrittelemämme syötteen paikkaan 0. Tämä kutsu ei ole välttämätön, ja jos se puuttuu, OpenGL antaa automaattisesti jokaiselle syötteelle oman sijainnin. On kuitenkin helpompaa, jos tiedämme etukäteen, mille paikoille syötteet päätyvät.

Nyt olemme melkein valmiita suorittamaan piirto-operaation. Tarvittavat tilat on kuitenkin vielä asetettava:

```
glVertexAttribPointer(POSITION,
    // Verteksissä on
    // kaksi komponenttia
    2,
    GL_FLOAT,
    // Ei normalisointia
    false,
    // Siirtymä tavuina
    // verteksien välillä
    sizeof(Vertex),
    vertices);
glEnableVertexAttribArray(POSITION);
glUseProgram(prog_id);
```

Ja lopulta varsinainen piirto:

```
glClear(GL_COLOR_BUFFER_BIT);
// Piirretään neljä verteksiä, alkaen
// indeksistä 0, eli taulukon alusta
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
```

Piirtokoodi sijoitetaan yleensä silmukkaan, joka huolehtii myös syötteen ja muiden tapahtumien käsittelystä. Mikäli käytettävä ikkunointikirjasto tarjoaa oman pääsilmukkansa, voidaan piirtämiseen käyttää ajastinta. Koska 3D-grafiikassa käytetään yleensä kaksoispuskurointia, voi myös olla tarpeen antaa käsky esittää puskurin sisältö ruudulla. Nettisivultamme löydät esimerkkiohjelman täydellisenä.

Shader-ohjelmointi — outoa salatiedettä?

Shader-ohjelmointi mielletään monesti jonkinlaiseksi mystiseksi taidoksi, jonka hallitsemiseksi tarvitsee olla sekä matemaatikko että graafikko. Todellisuudessa alkuun kuitenkin pääsee nopeasti, ja jo muutamalla koodirivillä voi saada upeita efektejä näytölle.

Teksti: Hannu Viitala Kuvat: Hannu Viitala, Visa-Valtteri Pimiä

Shader-ohjelmointi on parhaimmillaan hyvinkin helppoa ja palkitsevaa. Erilaisia grafiikka-algoritmeja voi omien kykyjen ja mielikuvituksen puitteissa kehittää käytännössä rajattomasti. Ei ole todellakaan pakko käyttää valmiita korkean tason grafiikkakirjastoja, jossa OpenGL ja shaderit on paketoitu jonkin "helppokäyttöisen" API:n taakse - käytännössä se usein vain rajoittaa omaa innovointia. Jos haluat ottaa laitteesi grafiikkapiiristä lähes kaiken irti, niin paras vaihtoehto on tehdä ohjelmasi suoraan OpenGL:n päälle ja tehdä myös shader-ohjelmat itse.

OpenGL ES

OpenGL on ohjelmoijalle näkyvä rajapinta, jolla käytetään tietokoneen grafiikkaprosessoria. OpenGL ES (GLES) on puolestaan sen supistettu versio, joka on tarkoitettu vähemmän tehokkaisiin laitteisiin, kuten matkapuhelimiin. Tässä artikkelissa keskitytään GLES:n shader-ohjelmointiin. GLES on käytössä lukuisilla eri alustoilla, kuten Symbian, Android, iOS, ja jopa Blackberry. Poikkeuksen tekee Windows Phone, jossa käytetään Direct3D-rajapintaa, ja sen myötä shadereissa HLSL-kieltä, joka poikkeaa jonkin verran toiminnaltaan ja syntaksiltaan GLES:ssä käytettävästä GLSL-kielestä.

Shader on pieni ohjelma, joka suoritetaan näytönohjaimen grafiikkasuorittimessa (GPU) koneen pääsuorittimen (CPU) sijaan. Grafiikkasuoritin kutsuu shadereita grafiikkaliukuhihnan tietyssä vaiheessa, jolloin ohjelmoija voi vaikuttaa lopputulokseen erittäin monipuolisesti. Erityyppisiä shadereita kutsutaan liukuhihnan eri vaiheissa. GLES 2.0 tukee shader-ohjelmointia kulmapistetasolla (vertex shader) ja pikselitasolla (fragment shader). Työpöytätietokoneissa käytettävä OpenGL sisältää myös geometriashaderin (geometry shader), mutta GLES 2.0:sta se puuttuu.

Monesti ohjelmoijalle tulee yllätyksenä, että GLES 2.0:n mukana ei tavallisesti tule mitään valmista shader-kirjastoa, josta valita sopivia ohjelmia, vaan kaikki pitää, ja ennen kaikkea saa, tehdä itse. Ilman shader-ohjelmia näytölle ei renderöidä yhtään pikseliä (lukuunottamatta taustaväriä). Aloituskynnys on tämän vuoksi korkeammalla kuin GLES 1.1:ssä, jossa shadereita vastaava toiminnallisuus on toteutettu kiinteänä funktioputkena. Vanhemmassa GLES-rajapinnassa on sisäänrakennettuna kehittyneitä toimintoja, kuten valonlähteen ja varjostustyypin valinta, mutta ne on karsittu pois 2.0:sta. Versiot eivät näin ollen ole keskenään yhteensopivia. Poistuneet toiminnot toteutetaan shader-ohjelmilla, joiden kautta ohjelmoija määrittelee itse grafiikkasuorittimen toimintaa eri tilanteissa.

Toinen yllätys on shaderien virheenetsintämahdollisuudet, jotka ovat kuin jäänne 80-luvulta tai jostain vielä kauempaa. Minkäänlaisiä debug-tulostuksia ei ole käytössä, vaan tiedot on kirjoitettava pikseleiksi! Jos ohjelmoija siis haluaa tietää shader-ohjelman sisäisen muuttujan arvon, se pitää koodata väriarvoksi ja tulostaa pikselishaderin avulla ruudulle.

Shaderien ohjelmointikieli, GLSL, on syntaksiltaan yksinkertaista. Se muistuttaa karsittua C-kieltä, johon on lisätty esimerkiksi valmiit tietotyypit mm. erilaisille vektoreille ja matriiseille. GLSL tarjoaa välineet todella monimutkaisten grafiikka-algoritmien toteutukseen, mutta myös yksinkertaisilla laskutoimituksilla, ja vaikkapa valmiiden trigonometristen funktioiden käytöllä, voi saada vaikuttavaa jälkeä aikaiseksi. Kaikki on lopulta kiinni omasta kekseliäisyydestä eikä välttämättä korkean tason matemaattisista taidoista.

GLES-shaderit

Kolmio on yksinkertaisin geometrinen muoto, joka muodostaa tason. Niinpä shadereissa kaikki objektit perustuvat yleensä kolmioihin (triangle), vaikka muitakin primitiive-

jä löytyy, kuten viiva (line) ja piste (point). Primitiivit määritellään kulmapisteinä eli vertekseinä. Neliö saadaan aikaiseksi, kun laitetaan kaksi kolmiota vierekkäin. Samalla periaatteella voidaan tehdä mielivaltaisen muotoisia kappaleita.

Grafiikkasuoritin ajaa renderöinnin yhteydessä kulmapisteohjelman jokaiselle kulmapisteelle. Kulmapisteohjelmassa voidaan muuttaa kulmapisteen koordinaatteja, mutta niiden lukumäärään ei voi vaikuttaa. Siihen tarvittaisiin geometriaohjelmaa, joita ei siis ole GLES 2.0:ssa. Kulmapisteohjelmalla grafiikkasuoritin laskee kulmapisteen lopulliset koordinaatit ruudulla.

Pikseliohjelmaa kutsutaan jokaiselle näytön pikselille, jossa saattaa olla jotain piirrettävää, eli silloin kun pikseli kuuluu johonkin rasteroitavaan primitiiviin. Tämän takia pikseliohjelmaa kutsutaan yleensä paljon useammin kuin kulmapisteohjelmaa, joten pikseliohjelman suorituskyky on paljon kriittisempi lopputuloksen kannalta. Pikseliohjelmassa asetetaan väriarvo ja peittävyys yksittäiselle pikselille.

Shader-ohjelmien tiedonvälitykseen käytetään kolmen tyyppisiä muuttujia: attribute, uniform ja varying. Tietoa voidaan välittää isäntäohjelman (esim. C++-ohjelma) ja shader-ohjelman välillä, eri shaderien välillä tai GPU:n ja shaderien välillä. Attribute-muuttuja on kulmapistekohtainen muuttuja, joka annetaan C++:sta taulukkona OpenGL API:lle. Jokaista kulmapistettä vastaa yksi taulukon alkio. Näin kulmapisteohjelmalle voidaan välittää vaikkapa koordinaatit, pinnan normaali, tekstuurikoordinaatit tai väriarvo jokaisessa kulmapisteessä. Attribute-muuttujia voi käyttää vain kulmapisteohjelmissa. Jos attribute-muuttujan arvoa tarvitaan myös pikseliohjelmassa, niin se pitää välittää kulmapisteohjelmalta pikseliohjelmalle varying-muuttujan avulla. Varying-muuttujan kautta annettu data interpoloidaan pikseliohjelmalle niin, että sen kulloinenkin arvo riippuu pikselin etäisyydestä jokaiseen kolmion kulmaan. Uniform-muuttuja taas pysyy vakiona piirtokomennossa annetulle joukolle kulmapisteitä ja niihin liittyviä pikseleitä. Usein käytettyjä uniform-muuttujia ovat animaatioissa tarvittava aikamuuttuja sekä kulmapistekoordinaattien muokkaamiseen käytettävät muunnos- ja projektiomatriisit. GLSL:ssä on myös sisäisiä muuttujia, kuten gl_Position, joka välittää kulmapisteen lopulliset koordinaatit GPU:lle.

Vielä yksi tapa välittää tietoa shadereille on tekstuuri. Yleensä se sisältää pikseliohjelmassa tarvittavaa väri-informaatiota, mutta se voi sisältää myös mitä tahansa numeerista tietoa, kuten pinnan normaaleja tai muunnosmatriiseja. Joissakin OpenGL-toteutuksissa tekstuurin kautta voi välittää tietoa myös kulmapisteohjelmalle.

Ensimmäiset omat shader-ohjelmat

Alla esiteltäviä pikseliohjelmia voi kokeilla käytännössä kirjoittamalla ne netissä oleviin shader-työkaluihin (linkit ohessa). Näissä työkaluissa ei yleensä voi muuttaa suoraan kulmapisteohjelmaa. Työkaluissa on käytettävissä myös valmiita uniform-muuttujia, kuten "time" ja "mouse", joilla saa helposti tehtyä vuorovaikutteisia animaatioita. Vastaavia shaderien kokeilutyökaluja löytyy sovelluskaupasta myös Android- ja iOS-laitteille.

GLSL Sandbox
http://glsl.heroku.com/
Shader Toy
http://www.iquilezles.org/apps/shadertoy/

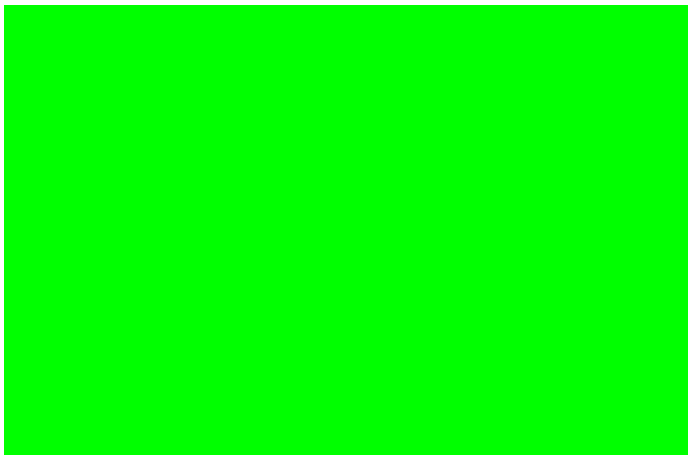
Kulmapisteohjelmassa **(listaus 1)** ei tarvita välttämättä kuin yksi attribute-muuttuja, joka määrittää kulmapisteen koordinaatit (vertexPos). Kulmapisteohjelma sijoittaa kulmapisteen lopulliset koordinaatit gl_Position-muuttujaan. Isäntäohjelmassa (jota ei esitellä tässä) alustetaan kulmapisteet niin, että ne muodostavat neliön, eli kaksi kolmiota vierekkäin.

Pikseliohjelma **(listaus 2)** on vieläkin yksinkertaisempi, sillä se ei välttämättä tarvitse mitään syötettä. Oheinen pikseliohjelma tuottaa vihreitä pikseleitä sijoittamalla kyseisen väriin arvon gl_FragColor-muuttujaan. Väri muodostetaan neljästä liukuluvusta, joiden arvot ovat väliltä 0.0 ja 1.0: punainen, vihreä, sininen ja peittävyys.

Seuraavaksi **(listaus 3)** kokeillaan jotain vähän mielenkiintoisempaa: muutellaan pikselin väriarvoa sen sijain-

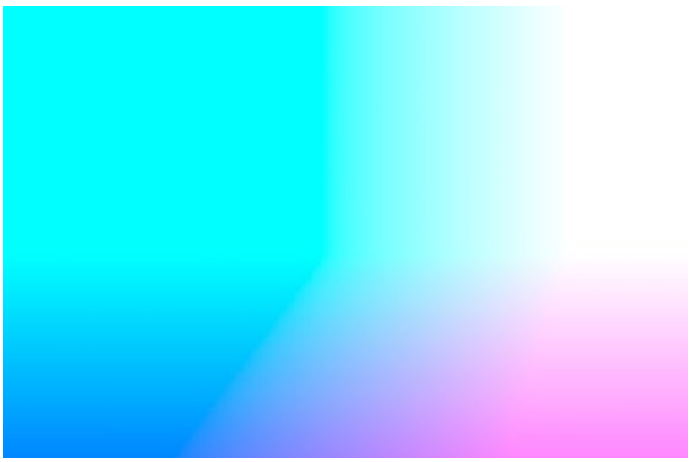
```
attribute vec3 vertexPos;
void main(void)
{
    gl_Position = vec4(vertexPos, 1.0);
}
```

Listaus 1



```
precision mediump float;
void main(void)
{
    gl_FragColor = vec4(0.0, 1.0, 0.0, 1.0);
}
```

Listaus 2 ja tuloksena syntyvä kuva

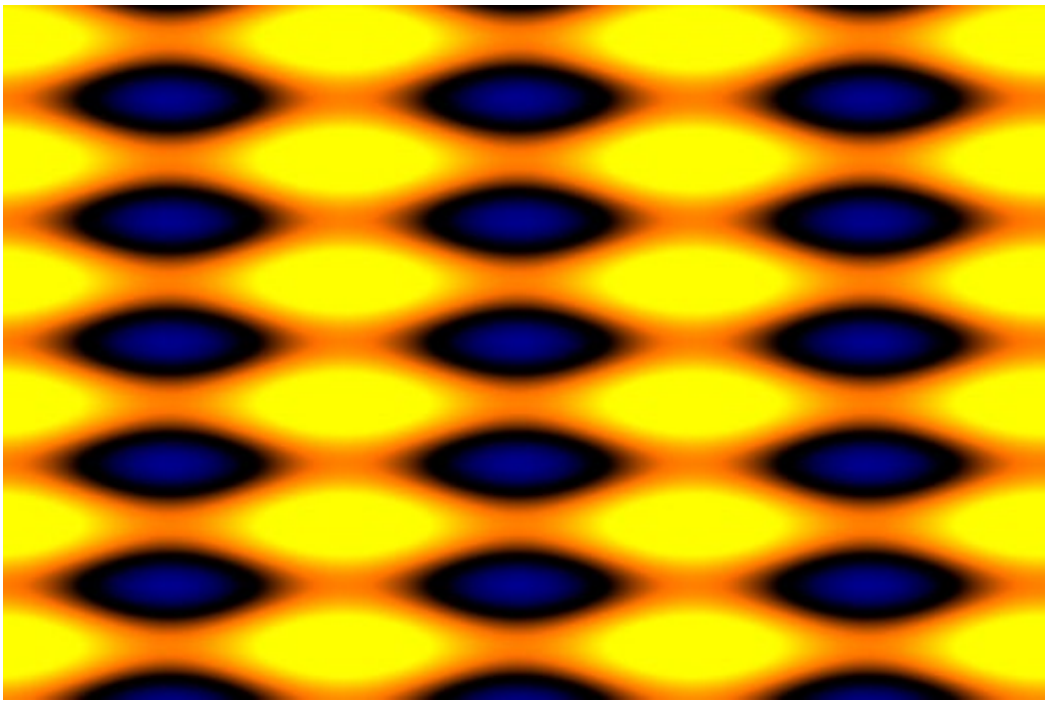


```
precision mediump float;
void main(void)
{
    vec2 pos = ( gl_FragCoord.xy / vec2(320, 240));
    gl_FragColor = vec4(pos.x, pos.y, 1.0, 1.0);
}
```

Listaus 3 ja tuloksena syntyvä kuva

nin mukaan. GLSL antaa pikseliohjelmalle kulloisenkin pikselin sijainnin gl_FragCoord-muuttujan kautta. Sijaintikoordinaatit skaalataan pienemmäksi, jotta ne soveltuvat r- ja g-väriarvoiksi. Yksittäisen värikomponentin maksimiarvo on 1.0, eikä sitä suuremmilla arvoilla ole enää vaikutusta lopputulokseen.

Listauksessa 4 on pikseliohjelma, joka piirtää näytölle värikkään salmiakkikuvion. Siinä käytetään gl_FragCoord-muuttujan lisäksi sini- ja kosinifunktioita pikselien väriarvojen laskemiseen. Pienellä muutoksella **(listaus 5)** voidaan shader-ohjelmassa ottaa käyttöön aikamuuttuja, jolla saadaan näytölle liikettä. (Muuttuja "time", kuten myös aiemmin esitelty "vertexPos", pitää erikseen välittää isäntäohjelmasta shaderille).



```
precision mediump float;
void main( void )
{
    vec2 pos = ( gl_FragCoord.xy / vec2(320, 240));
    float color_r = 1.0 + sin( pos.y * sin( 1.0 ) * 40.0 ) +
                          cos( pos.x * sin( 0.4 ) * 40.0 );
    gl_FragColor = vec4( color_r, color_r * 0.5,
                         sin( color_r + 10.0 / 3.0 ) * 0.75 , 1.0 );
}
```

Listaus 4 ja tuloksena syntyvä kuva

```
precision mediump float;
uniform float time;
void main( void )
{
    vec2 pos = ( gl_FragCoord.xy / vec2(320, 240));
    float color_r = sin(time) + sin( pos.y * sin( 1.0 ) * 40.0 ) +
                    cos( pos.x * sin(  0.4 ) * 40.0 );
    gl_FragColor = vec4( color_r, color_r * 0.5,
                         sin( color_r + 10.0 / 3.0 ) * 0.75 , 1.0 );
}
```

Listaus 5

Miten tästä eteenpäin?

Tässä artikkelissa on vain raapaistu pintaa siitä mitä shadereillä voi saada aikaiseksi. Seuraavaksi kannattaa tutustua vaikkapa muunnosmatriiseihin, tekstuureihin, pinnan normaaleihin, ja erilaisiin 3D-varjostusalgoritmeihin.

Kokonainen ajettava ohjelma sisältää shader-ohjelmien lisäksi tietenkin OpenGL-rajapintaa käyttävän rungon, jonka voi tehdä esimerkiksi C++:lla, Pythonilla, Javalla tai Objective-C:llä. Tällainen ohjelma on esitelty tässä lehdessä toisaalla. Hyviä OpenGL-esimerkkejä on myös sivulla http://nehe.gamedev.net/ Virallinen GLES 2.0:n määrittely löytyy Khronoksen sivulta: http://www.khronos.org/

Vaikka OpenGL-rajapinnan avulla on mahdollista tehdä alustariippumatonta koodia, joka kääntyy suoraan eri ympäristöihin, tarvitaan kuitenkin aina myös alustakohtaista koodia, kuten kuvatiedostojen lataaminen muistiin, OpenGL:n alustus, syöttö- ja tulostuslaitteiden käsittely jne. Tätä silmälläpitäen on tehty kirjastoja ja sovelluskehyksiä, joilla alustariippuvuudet saadaan minimoitua, kuten SDL, Qt ja Cocos2dx. Lisäharmia aiheuttaa esim. Android-ympäristössä se, että laitteet ovat toteutukseltaan ja ominaisuuksiltaan erilaisia. Puhelinvalmistajilla on erilaisia grafiikkasuorittimia ja ajuriohjelmia. Ajurien laatu ja toiminta saattaa vaihdella merkittävästi jopa saman valmistajan eri malleissa. Esimerkiksi jossain HTC:n laitteessa voi valita vapaasti tekstuurin korkeuden ja leveyden, kun taas tietyissä Samsungin laitteissa niiden on oltava kakkosen potensseja. Eroja on myös siinä, mitä OpenGL-laajennuksia laite tukee. Tällaisia laajennuksia ovat muun muassa tekstuuriyksikön käyttö kulmapisteohjelmassa ja tekstuuripakkausalgoritmit. PC:tä käytetään paljon ajamiseen ja testaamiseen kehitysvaiheessa, vaikka lopullinen ohjelma tulisikin matkapuhelimeen tai tablettiin. PC:llä voi kuitenkin olla käytössä enemmän shader-muistia kuin mobiililaitteessa, mikä aiheuttaa yhteensopivuusongelmia.

Kaiken kaikkiaan shader-ohjelmoinnissa on kuitenkin samaa löytämisen riemua joka syntyi, kun aikoinaan siirtyi Basicista ohjelmoimaan 6502-konekieltä, ja huomasi millainen tehoreservi olikin yhtäkkiä käytössä. Sen jälkeen ei ollut enää paluuta Basiciin. Samoin, kun on päässyt sisälle shader-ohjelmointiin, pääsuorittimella toteutettu grafiikka tuntuu toivottoman hitaalta. Shader-ohjelmointi on niin paljon helpompaa, tehokkaampaa ja ennen kaikkea hauskempaa.

Perusviileä yhteisö

CoolBasic on suomalainen, pelien tekemiseen suunnattu aloittelijaystävällinen ohjelmointikieli. Sen ympärillä toimii aktiivinen yhteisö, jonka jäsenet järjestävät tapaamisia, kilpailuja ja muita yhteisiä projekteja. Säännöllisesti toimintaan osallistuvaa väkeä on viitisenkymmentä, ja vähemmän aktiivisia osanottajia useampi sata.

Teksti: Olavi Lintumäki Kuvat: Olavi Lintumäki, Ville Lahdenvuo

Koska CoolBasic on ominaisuuksiltaan rajoittunut, se harvemmin houkuttelee kokeneita ohjelmoijia. Aloittelija taas pääsee sen parissa helposti alkuun kattavan suomenkielisen manuaalin ja yhteisön avulla. Kieltä on helppo lähestyä, sillä näkyviä tuloksia saa aikaan verrattain nopeasti. Vaikka suuri osa harrastajista lähtee ennen pitkää kehittämään osaamistaan uusiin suuntiin, CoolBasic säilyttää erityisen paikkansa monen sydämessä.

CoolBasic sopii nopeaan testailuun ja prototyyppien kehittämiseen. Monien algoritmien toteuttaminen on vaivatonta, vaikka kieli ei tarjoa valmiina taulukoita monimutkaisempia tietorakenteita, eikä kehittyneempien tietorakenteiden toteuttaminen sillä ole aina luontevaa. Osoitinten puutetta paikkaavat itse määriteltävät tietuetyypit, joihin voi viitata, sekä muistipalat, jotka ovat lähimpänä C-tyylisiä taulukoita.

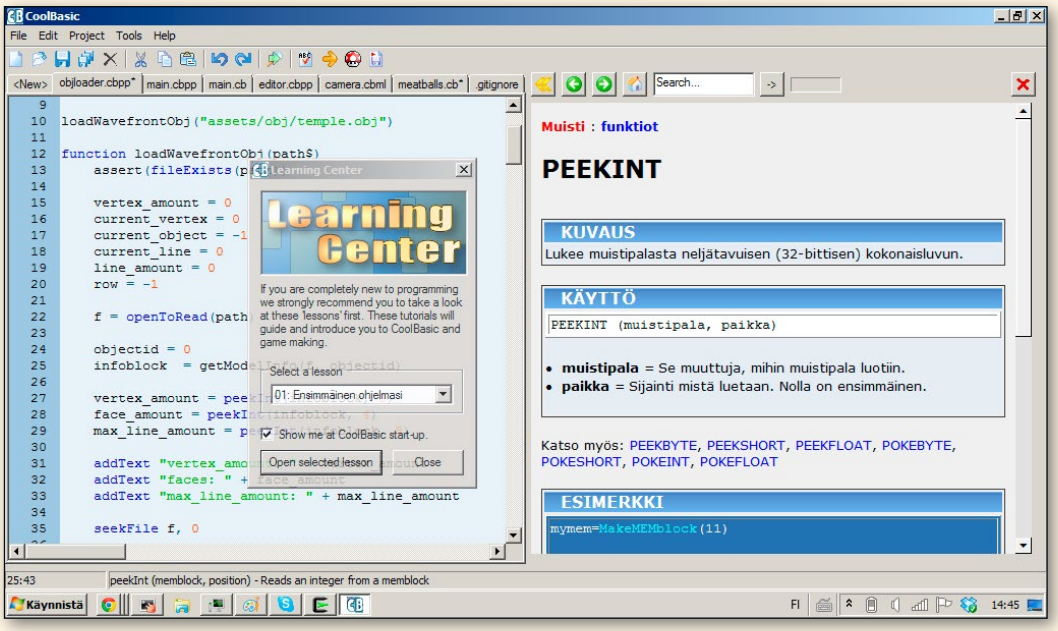
Motivoivia haasteita

CoolBasicin nopeus on nykymittapuulla toivoton, mutta se ei harrastajia haittaa. Hitaus toimii pikemminkin haasteena, joka motivoi nopeampien algoritmien ja tekniikoiden löytämiseen. Vaikuttavan lopputuloksen tavoittelu on mielenkiintoista, kun tehot eivät riitä mielivaltaiseen grafiikkaan tai monimutkaiseen prosessointiin. Silloin tällöin löytyy uusia tapoja kiertää tai hyödyntää rajoituksia, mikä avaa mahdollisuuksien rajaseuduille uusia, tutkimattomia alueita. Aina voi kilpailla yhä paremmista suorituksista ja pyrkiä tekemään jotain sellaista, mitä CoolBasicilla ei ole vielä nähty.

Suurin pullonkaula suorituksessa on virtuaalikoneen nopeus. Vauhtia on haettu esimerkiksi kuvista, viivoista ja muista ruutua kerralla enemmän täyttävistä piirtokomennoista. Myös resoluutio on luonnollisesti pysynyt varsin alhaisena ja esilaskenta on ollut suosittua. Erityisesti

Mikä CoolBasic?

- Jukka Lavosen (zerppa) lukioikäisenä kehittämä, välikielestä tulkattu ohjelmointikieli
- Esikuva ja syntaksin alkuperä on Blitz Basicissa
- Ensimmäinen versio vuodelta 2001, viimeisin beta 10.43 vuodelta 2005
- Keskustelualueella yli tuhat rekisteröitynyttä käyttäjää
- Eri vaiheisiin edenneitä pelejä on yli 400, introja ja demoja yli 40



Compoentryjen läpikäymistä

Scroller CBE

läpinäkyvyysmaskien avulla on mahdollista kehitellä uusia efektejä, joihin tehot eivät muuten mitenkään riittäisi. Vaikka pullonkaulat ja kyvyt ovat jakautuneet eri tavoin, on kokonaisvaikutelma silti jossain mielessä samantyyppinen kuin menneiden vuosikymmenten alustoilla. Tämä näkyykin lopputuloksissa positiivisella tavalla.

Miksi sitten käyttää nyky-PC:llä ohjelmointikieltä, joka paikoin häviää 80-luvun tietokoneillekin? Jos kerran vähempikin teho riittää, miksei saman tien käytä vanhoja tietokoneita? Eräs harrastaja tiivistää asian seuraavasti: ”CoolBasic on minun Commodoreni.” Yksinkertaisesta ohjelmointikielestä on kehittynyt käyttäjilleen omalta tuntuva, erityinen alustansa. Nostalgiasta on tässä tapauksessa liian varhaista puhua, sillä suurin osa tekijöistä on ollut menossa mukana alusta saakka, eikä CoolBasic ole kuin vasta reilut kymmenen vuotta vanha.

Nopeutta kehiin

Skeneä järisyttää parhaillaan vaihtoehtoinen yhteisölähtöinen virtuaalikone ja runtime, cbEnchanted. Se on alkuperäistä nopeampi ja laajentaa grafiikkaominaisuuksia merkittävästi. Sen tekemiseen ryhdyttiin, koska CoolBasicin uutta versiota ei ole kuulunut eikä nykyversio toimi Linuxilla.

Enchantedin mahdollisuuksien rajat ovat luultavasti huikeat. Sen testailu on toistaiseksi jäänyt melko vähäiselle tasolle, sillä se on osoittautunut joiltakin osin alkuperäistä heikommaksi ja on ollut käytettävässä kunnossa vasta vähän aikaa. Samalla kun Enchanted tarjoaa uusia ominaisuuksia ja mahdollisuuden luoda niitä hyödyntäviä efektejä, joutuu sen kanssa luopumaan joistain totutuista tempuista ja keksimään tilalle aivan uusia.

Jatkossa nähdään varmasti molemmilla versioilla tehtyjä tuotoksia, sillä myös alkuperäisen kielen potentiaalista on vielä paljon käyttämättä. Toistaiseksi molemmilla versioilla tehdyt tuotokset kilpailevat samassa sarjassa, mutta saattavat erottautua tulevaisuudessa selkeästi omiksi alustoikseen. Enchanted on joka tapauksessa jo nyt tuonut CoolBasic-skeneen uutta intoa pelien ja demojen tekemiseen.

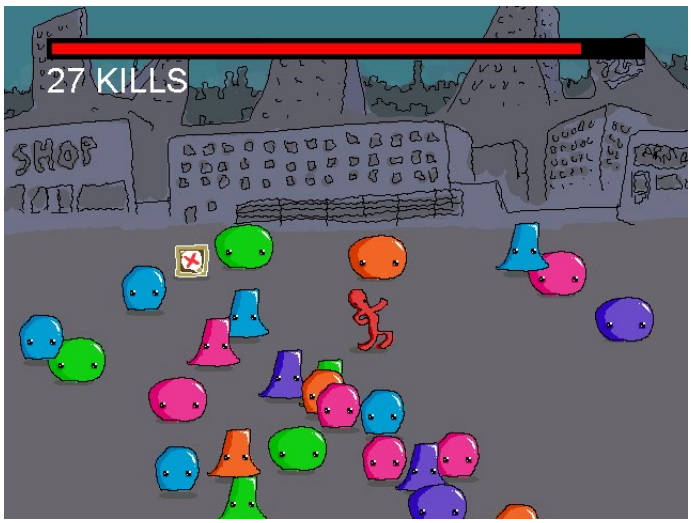
Laadukasta lanitusta

Yhteisön tapaamiset ovat jakaantuneet etäisyyksien ja mieltymysten perusteella muutamaan eri luokkaan. Vanhempi, pidempään mukana ollut porukka käy vuosittain mökillä nauttimassa kesästä rauhallisissa merkeissä. Laneilla on vilkkaampaa, ja Assemblyillä vietetään party- ja tiiviissä tunnelmissa omien kilpailujen ja tempausten säestämänä.

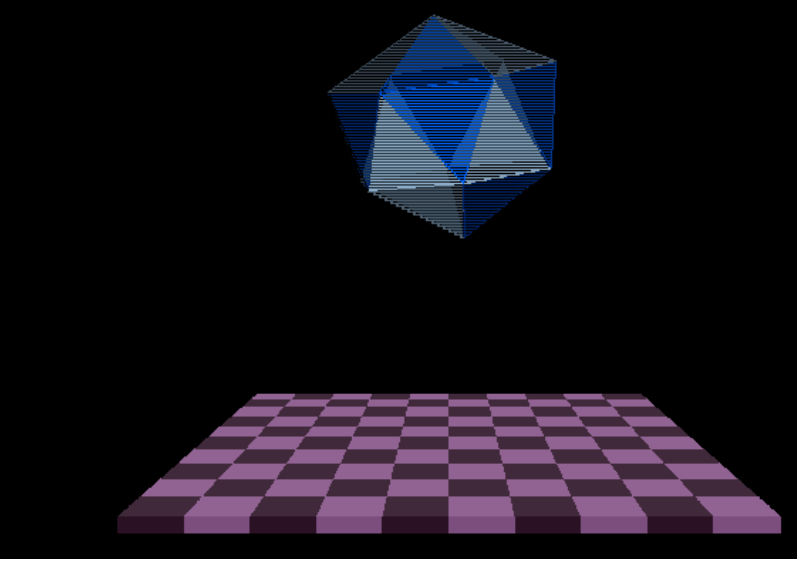
Tavallisista laneista CoolBasic-kokoontumiset eroavat siinä, että suuri osa ajasta kuluu pelaamisen lisäksi koodaamiseen joko keskinäisten kilpailujen tai yhteisten ja omien projektien muodossa. CoolBasicilla tehtyjen pelien pelaaminen on yleistä, ja kilpailuihin osallistutaan yleensä sillä tehdyillä tuotoksilla, mikäli tämä sopii kilpailun laatuun. Laneja on järjestetty viime vuosina säännöllisemmin, ja kompoja on ollut kerralla enimmillään yli viisi.

Assemblyilla CoolBasic-ryhmä on nähty säännöllisesti vuodesta 2007 lähtien. Se valtaa mielellään tilavat käytäväpaikat itselleen mahtuakseen paremmin liikkumaan koneelta toiselle, tekemään yhteistyötä ja seuraamaan muiden tekemisiä. Messuhal-

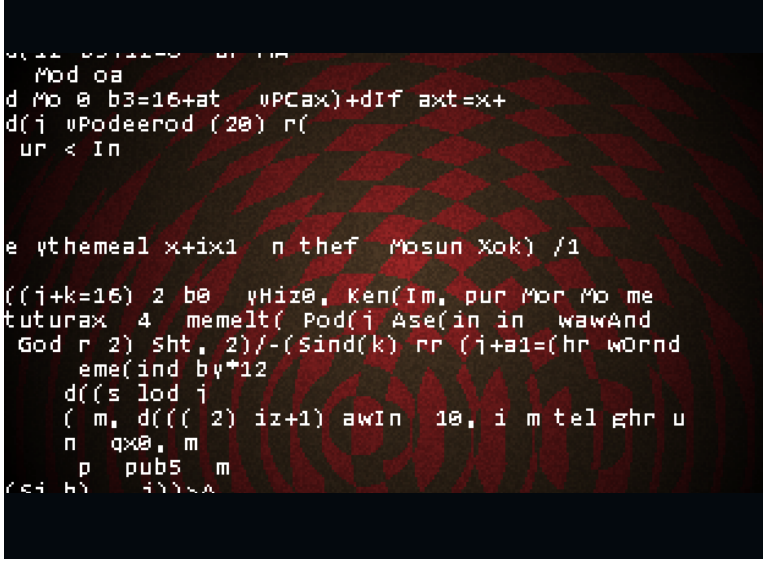
Tulipalo



Turbo Jelly



White Noyce



Linear Minds

lin ulkopuolella järjestetään virallinen tapaaminen, johon osallistuu myös CoolBasicin kehittäjä Jukka Lavonen.

Ensimmäisellä kerralla tapaamisessa pidettiin muutaman tunnin pelikilpailu, jota on seurannut liki sata viikko- tai pikakisaa. Kilpailujen ideana on ohjelmoida peli rajatussa ajassa, ja niitä värittää aiheen tai sallittujen tekniikoiden rajaus. Ensimmäisessä kilpailussa piti tehdä 30x20 pikselin ruudulla pelattava peli. Voittaja on aina valittu äänestämällä.

Muuten Assemblyilla on keskitytty enemmän virallisiin kilpailuihin osallistumiseen. CoolBasic-demojen julkaisu ei ole koskaan osunut Assemblyille, vaan foorumeilla järjestetyt kilpailut ovat päättyneet yleensä talvella tai alkukesästä.

Introja ja demoja

Ensimmäinen arkistoitu introkisa järjestettiin vuonna 2007. Rajoitteet liikkuivat aluksi muutamassa sadassa rivissä koodia, ja noin megatavussa muuta mediaa. Tämän vuoksi on tapana julkaista aina myös koodi, vaikka muutamassa viimeaikaisessa kilpailussa rajoituksia ei ole enää ollut. Rajoitusten löyhdyttyä on alettu puhua introkilpailun sijaan demokilpailusta.

Voi olla, että jossain vaiheessa järjestetään taas rajoittuneempia kilpailuja, mutta tällä hetkellä yhteisö on kiinnostuneempi etsimään alustan todellisia rajoja. Kenties jonkinlainen vuorottelu toimisi parhaiten. Yhteisön pienen koon vuoksi segmentointia ei voi tehdä loputtomiin, vaan on keskityttävä siihen, mikä luo eniten kilpailua. Samasta syytä ryhmäytyminen on ollut hidasta, vaikka sitä onkin ollut ilmassa vaivan määrän kasvettua rajoitusten vähentyessä. Ryhmittymät joutuvat muovautumaan sen mukaan, miten suurella panoksella kukin ehtii osallistua kilpailuihin.

Onko CoolBasicilla tulevaisuutta?

CoolBasicin kehitys on näennäisesti pysähtynyt vuoteen 2005, jolloin viimeisin virallinen versio julkaistiin. Sen jälkeen kehitystyö on aloitettu alusta uudella tiimillä useampaan otteeseen, mutta valmista ei ole tullut, sillä vapaa-ajalla on vaikea tehdä täydellistä tuotetta. Näyttää siltä, että aiemmasta oppineena kehitystiimi on nyt painunut maan alle, eikä lupaile julkaisua ennen kuin se on oikeasti käsillä.

Tulevaan versioon on suunniteltu ja suurelta osin jo toteutettu kehitysympäristö, moderni virtuaalikone ja pelien tekemiseen soveltuva nykyaikaisempi runtime. Tämä versio ei olisi rajoittunut yhteen runtimeen, vaan sitä voisi tarpeen mukaan vaihtaa. Ympäristö ei myöskään olisi myöhemmissä versioissa rajoittunut basiciin, vaan siinä olisi mahdollisuus kääntämiseen useammalta kieleltä. Myös yhteisöllisiä ja opetuskäytöllisiä ominaisuuksia on suunniteltu, sillä niille on selvää kysyntää.

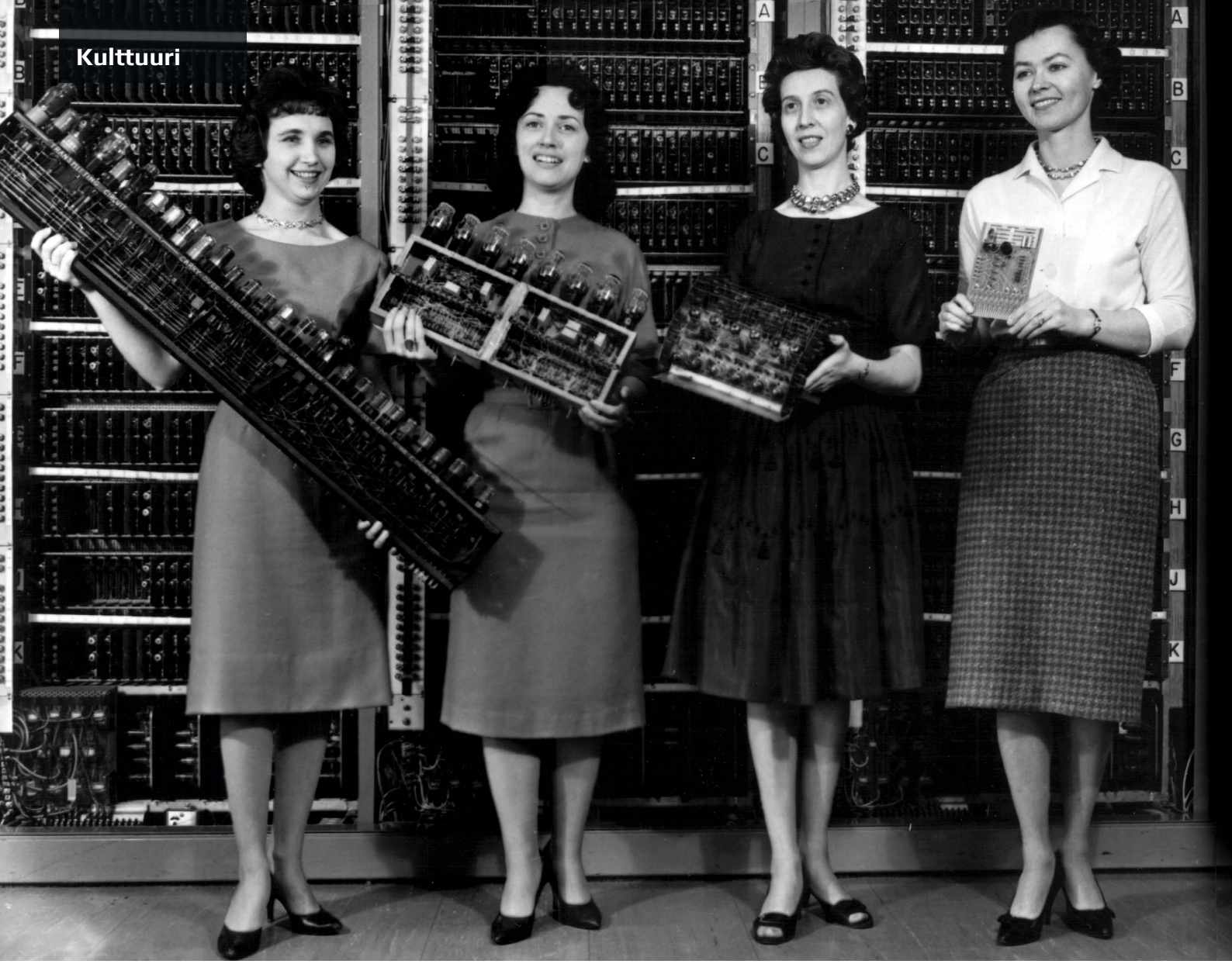
Voi olla, että Lavosen lukioajan projektista on kasvanut liian suuri urakka vapaa-ajallaan työskentelevälle kehitystiimille. Huhu kertoo, että kehitystä tapahtuu hiljakseen koko ajan, mutta siitä ei kerrota paljoa. Seuraavaa versiota odotellessa ei siis voi muuta kuin jatkaa vanhaan malliin demoja, pelejä ja omia täydennyksiä CoolBasiciin tehden. Ei ole mikään katastrofi, jos uutta versiota ei tulekaan, sillä nykyisen kielen piirteiden kanssa on opittu elämään. Olisi kuitenkin mahtavaa, jos kotoisa ja kotimainen CoolBasic voisi vihdoin liittyä vakavasti otettavien ohjelmointikielten joukkoon.

```
alku=timer()
Repeat          ’pääsilmukka, toistuu monta kertaa sekunnissa luoden animaation
        aikanyt=Timer()-alku
        // satunnaislukugeneraattori alustetaan jokaisella kierroksella,
        // jotta tähdelle arvotaan aina sama paikka
        Randomize 5680111
        // käydään läpi tähdet
        For i=0 To 1024
                d=(aikanyt*0.04+i) Mod 1024             ’lasketaan pisteelle etäisyys
                // mod saa aikaan sen, että liian kaukana olevat tähdet hyppäävät alkuun
                c= min(255,1024.0/d)                    ’siitä saadaan kirkkaus näin
                Color c,c,c                             ’asetetaan se väriksi
                //arvotaan sijainti ruudulla ja jaetaan se lasketulla etäisyydellä
                Box 200+Rnd(-5000,5000)/d,150+Rnd(-5000,5000)/d,c*0.1,c*0.1
                // viimeiset parametrit edustavat tähden kokoa,
                // joka tulee sopivasti kirkkaudesta
        Next i
        DrawScreen      ’asettaa äsken piirretyt tähdet näkyviin joka kierroksella
Forever ’seuraava kierros
```

Koodiesimerkki: Tähtitaivaanvieritin. Menee takaperin, mutta lukija saa korjata jos niin tykkää.

Näin yhteisö ehostaa CoolBasicia

- Ominaisuuksien laajentaminen lisäkirjastoilla: muun muassa OpenGL- ja chipmunk-wrapperit ja muutama verkkokirjasto.
- cbEnchanted on vaihtoehtoinen, tehokkaampi ja kyvykkäämpi virtuaalikone ja runtime, jossa on myös järkevämpi kirjastotuki.
- Muutama kääntäjä- ja esikääntäjäprojekti on ollut aluillaan, mutta suuria tuloksia ei vielä ole.



Yhdysvaltain puolustusvoimien varhaisia tietokonetyöläisiä. Vasemmalta oikealle: Patsy Simmers, Gail Taylor, Milly Beck ja Norma Stec.

Tietokonekulttuuri ja sukupuoli

Tietokoneet ovat olleet miesten harrastus viimeiset parikymmentä vuotta. Alan sukupuolijakauma on kuitenkin aiemmin ollut tasaisempi, ja muutosta on varmasti luvassa myös lähitulevaisuudessa.

Teksti: Ronja Koistinen Kuvat: Yhdysvaltain puolustusvoimat, Anita Sarkeesian

Mikrotietokoneiden ja videopelien ympärille alkoi kehittyä omaa alakulttuuria viimeistään 1980-luvulla. Koteihin ostettiin televisioihin kytkettäviä viihdelaitteita ja tietokoneita. Näiden suurimmaksi käyttäjäryhmäksi muodostuivat syystä tai toisesta pojat. Erityisesti koneista kaiken riemun irti ottaminen ja niihin korvia myöten uppoutuminen vaikuttaa olleen poikien ja miesten ajanvietettä. Tietysti intohimoisesti Zeldaa pelanneita tyttöjä ja silloin tällöin purkkeihin soitelleita naisia on ollut varmasti useitakin, mutta kirjahyllyittäin pelejä keränneet ja demokoodaamiseen hurahtaneet ovat lähes kaikki miehiä.

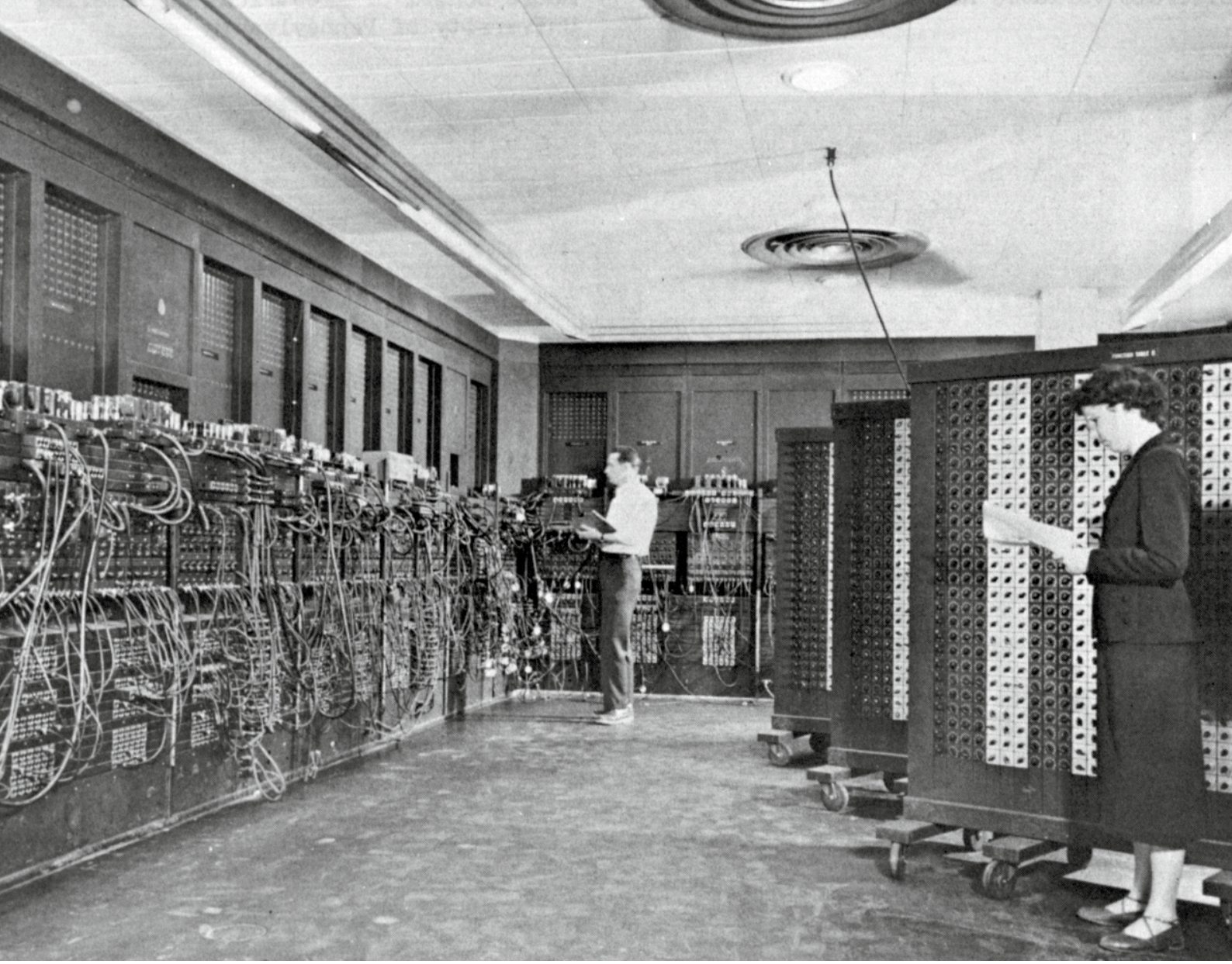
Niin, ja kun kerran luet tätä lehteä, olet hyvin todennäköisesti mies. Tätä kirjoitettaessa Skrollin tilaajista etunimen perusteella arvaten vain noin 2,5 % on naisia. Luku on tyrmäävä, muttei kovin yllättävä. Skrollin IRC-kanavalla tilanne ei mittaushetkellä huhtikuussa ollut aivan yhtä jäätävä: käyttäjistä noin 4 % oli naisia.

Demoskenen äijät

Tietokonekulttuuri on nykyään miehistä, ja demoskene on tästä ehkä raain esimerkki. Skenessä miehet kohtaavat yhteisen toiminnan merkeissä, arvostavat toisiaan meriittien mukaan ja mittelevät paremmuudesta ohjelmointikikkojen areenalla. Demopartyt ovat kuin jonkinlaiset turnajaiset, joihin naiset osallistuvat harvoin muuten kuin sivustakatsojina.

Miehillä on kenties naisia useammin tapana kokoontua vapaa-ajallaan jonkin yhteisen touhun ympärille olemaan tosissaan. Kyse voi olla tietokoneista, jalkapallosta tai vaikka mopojen virittelystä. Miesten keskinäisessä harrastelussa on tavallista, että joitain asioita vetäydytään tekemään syrjään äijäseurassa, ja naisten osallistumiseen suhtaudutaan parhaimmillaankin vaihtelevasti.

Feministisessä sukupuolentutkimuksessa on käytetty miesyhteisöistä ja miesten välisestä ystävyydestä termiä homososiaalisuus. Sillä tarkoitetaan (hetero)miehille ominaista tapaa olla yhdessä toisten miesten kanssa. Eräs kiinnostava malline on ns. homososiaalinen objekti, jokin konkreettinen sosiaalisten suhteiden ulkoinen katalyytti, jonka



Glen Beck (vas.) ja Betty Snyder (oik.) työskentelemässä ENIAC-tietokoneen kanssa 1940- tai 1950-luvulla.

äärelle miehet tykkäävät kokoontua. Tällainen voi olla vaikkapa jääkiekko, olut tai autot.[1]

Myös demoskeneharrastusta on kiinnostavaa ajatella homososiaalisuuden kautta. Kilpaileminen, meritokraattiset hierarkiat ja kunnioitusjärjestelmät ovat vahvasti läsnä homososiaalisissa miesyhteisöissä. Feimi ja greetzit ovat skenessä peruskauraa. Homososiaalisen käyttäytymisen ydinmotivaatio on tutkijoiden mukaan nimenomaan paikan löytäminen miehenä miesten joukossa. Tämä voi olla iso osatekijä siinä, miksi naisten on niin vaikea päästä sisään demoskenen kaltaisiin tietokonekulttuurin muotoihin.

Vielä kymmenen vuotta sitten Assemblyilla ei naisten vessaan tarvinnut juuri jonotella. Sittemmin tilanne on kuulemma muuttunut, ja Hartwall Areenan demotapahtuman kävijöistä tyttöjä ja naisia on ilmeisesti kymmeniä prosentteja joka vuosi. Tälle ei varmaankaan ole mitään yhtä tiettyä syytä, mutta muutos liittynee siihen, että Assemblyn kohderyhmä on pikkuhiljaa monipuolistunut pelkästä demoskeneyleisöstä laajempaan digitaalisen viihteen ja taiteen harrastajakuntaan.

Tympeää sihteerien työtä

Monesti esitetään kyynisiä analyysejä siitä, että noin 1970-luvulle asti tietokoneohjelmointia pidettiin tylsänä sihteerintyönä. Siis niin kauan kuin firmalla tai yliopistolla oli yksi tai kaksi tietokonetta, joita pääsi käyttämään vain pari ihmistä kerrallaan. Ohjelmointia verrattiin lähinnä konekirjoittamiseen. Matemaatikko, tilastotieteilijä, ballistikko tai muu arvostetussa asemassa oleva organisaation jäsen luonnosteli suunnitelman siitä, mitä tietokoneella piti laskea. Tämä suunnitelma kirjoitettiin ruutupaperille jossakin ihan muualla kuin tutkimuslaitoksen tietokonekellarissa, jossa ohjelmoijat työskentelivät. Suunnitelma luovutettiin sihteerin kaltaiselle ohjelmoijaolennolle, joka sitten muunsi saamansa matemaattisen ohjeistuksen tietokoneen ohjelmakoodiksi.

Työvaihetta ei pidetty tärkeänä tai erityisen arvostettavana, pikemminkin triviaalina ja mekaanisena. Alkuun näin ehkä tavallaan olikin, sillä varhaiset tietokoneet olivat varsin yksinkertaisia ja rajoittuneita kapistuksia, jumppasalin kokoisia taskulaskimia.

Alkuaikojen vähän arvostetut ohjelmoijat olivat huomattavan usein naisia. Esimerkiksi lämmöllä ja kunnioituksella muistellun Yhdysvaltain puolustusvoimien ENIAC-tietokoneen ohjelmoijista suurin osa oli naisia. Erikseen täytyy mainita yhdysvaltalainen amiraali Grace Hopper, joka kehitti muun muassa ensimmäisen kääntäjän digitaaliselle tietokoneelle, ”debugging”-termin sekä COBOL-ohjelmointikielen.

Kyynisen historiantulkinnan kautta voi arvella, että naiset on sysätty ohjelmointialalta syrjään siinä vaiheessa, kun koneet ja niiden mahdollisuudet ovat monipuolistuneet. Miehet valtasivat ohjelmoinnin, kun kävi ilmi, että kyseessä

Anita Sarkeesian julkaisee internetissä popkulttuurianalyysiä. Hänen viimeisin projektinsa käsittelee naiskuvia videopeleissä.

voikin olla sangen kannattava ja kunnianhimoinen, kasvava bisnes. Miesten työt kun ovat aina tärkeitä, ja kaikesta tärkeästä tulee miesten työtä![2]

Ohjelmoiminen on tänä päivänä edelleen miesvaltainen ala. Esimerkiksi Helsingin yliopiston tietojenkäsittelytieteen laitokselle vuoden 2012 päävalinnassa hyväksytyistä 26 % oli naisia.[3]

Keskustelu yltyy

Viimeisen vuoden aikana on puhuttu paljon naisista paitsi ATK-hommissa, myös videopelien pelaajina ja niiden kuvastoissa. Äänekkäimpänä on vellonut keskustelu kanadalais-amerikkalaisesta mediakriitikosta nimeltä Anita Sarkeesian. Hän on jo joitakin vuosia tehnyt Youtubessa varsin laadukasta feinististä analyysiä erilaisista median ilmiöistä Feminist Frequency -kanavallaan.

Keväällä 2012 Sarkeesian päätti ottaa tarkasteluun laajan otannan videopelien naiskuvastosta. Hän perusti Kickstarter-projektin Tropes vs. Women in Video Games (Kliseet vastaan naiset videopeleissä).[4] Joukkorahoitusprojektin sivulla pyydettiin yleisöltä lahjoituksia videosarjan toteuttamiseen. Sarkeesian asetti Kickstarteriin projektinsa rahoituksen päämääräksi 6000 dollaria.

Projektin julkistaminen sai yllättäen aikaan silmittömän viharyöpytyksen. Sarkeesian sai vihaista sähköpostia, törkeitä itseään esittäviä kuvamanipulaatioita sekä raiskaus- ja tappouhkauksia. Myös hänen yhteystieto'jaan yritettiin etsiä ja levittää.

Näiden äärimmäisten reaktioiden lisäksi myös projektin sanomaa vähäteltiin loputtomasti. Nettikeskusteluissa julistettiin kovaan ääneen, että ”feministit vihaavat kaikkea hauskaa”.

Vastareaktiona vihakampanjalle Sarkeesian sai kuitenkin tuntuvan suosionosoituksen samanmielisiltä tukijoilta: Kickstarter-kampanja sai nopeassa tahdissa rahoitusta yli 150 000 dollaria, monikymmenkertaisesti alkuperäisen tavoitesummansa verran.

Tropes vs. Women on viime kesästä muhinut pitkään ja hartaasti, koska valtavaksi yllätykseksi paisunut budjetti on pakottanut miettimään kaiken huolella uusiksi. Sarjan ensimmäinen osa julkaistiin vasta 7.3.2013. Vastaanotto on ollut vakuuttunutta: monet kommentoijat ovat jääneet odottamaan jatkoa innolla.

Tällä välin monet muut ovat ehtineet kommentoida naisten kuvauksia videopeleissä. Esimerkiksi Rock Paper Shotgun -verkkojulkaisu on kirjoittanut viime aikoina useita kertoja naisista sekä videopelien pelaajina että sisältönä, samoin The Escapist -sivuston vihaisen sarkastinen Jimquisition-videoblogi.

Keskustelussa liikkuu paljon kuumia tunteita joka suuntaan, ja välillä sekä lukijoiden että kirjoittajien on vaikea pysyä asiassa. Kiihkoilun lomasta pitää kuitenkin löytää oikeita ilmiöitä, koska muuten asiat eivät etene, vaan jäädään jumiin näköalattomaan kinasteluun.

Jotain keskeistä on jo selvästi saavutettu. Olennaisinta ja juurevinta on ollut muun muassa karkeistettu huomio siitä, että yleensä valtavirtavideopeleissä kaikki hahmot ovat fantasioita heteromiespelaajille. Naishahmot ovat seksifantasioita ja mieshahmot voimafantasioita miehille. Moni peli tarjoilee hahmot pelaajille sellaisista asetelmista, joissa toiminta-elementit ja pelimekaniikka välittyvät miespuolisten hahmojen kautta (usein suorastaan silmien takaa kuvattuna), mutta naiset ovat pelissä paljon passiivisemmin läsnä, katseen kohteina.

Videopelien yksinomaiselle miespelaajien liehittelylle on tietysti syynsä. Kalleimpien ja korkeaprofiilisimpien valtavirtapelien pelaajat ovat enimmäkseen miehiä. Kohderyhmäajattelu ohjaa tuotekehittelyä millä tahansa alalla.

Näivettävät itsestäänselvyydet

Videopelit ovat yhä näkyvämpi ja suurempi osa elämää yhteiskunnassamme. Ne ovat taidetta, ilmaisua ja kokemusta siinä missä kirjallisuus ja elokuvatkin. Siksi horisonttien avartaminen on tärkeää. Videopelit ovat älyttömän siisti juttu! On sääli, jos niissä ei ole mielekästä tarttumapintaa kokonaisille ihmisryhmille, kuten vaikka tytöille ja naisille. Naiset eivät lukumäärällisesti ole mikään vähemmistö pelaajien keskuudessa: vuonna 2012 videopelien kuluttajien kokonaismäärästä 47 % oli naisia tai tyttöjä.[5]

Silti ei tunnu mahdottoman virheelliseltä sanoa, että naiset eivät ehkä pelaa niin laajaa kirjoa erilaisia pelejä. Tarkkaa tietoa eri genrejen sukupuolittumisesta on vaikea löytää, mutta naiset pelaavat nykyään paljon ainakin MMORPG- ja muita roolipelejä, joissa voi mielensä mukaan tuunata omanlaisensa roolihahmon. Itse ainakin viihdytyn paljon välittömämmin pelistä, jossa voi valita naispuolisen pelihahmon. Riittää jo vaikka Nethack-tasoinen täysin abstrakti mielikuvataso; pelkkä vaihtoehdon olemassaolo on kiva juttu.

Naiset ja tytöt pelaavat myös The Simsin kaltaisia pelejä, joissa niin ikään on laajat mahdollisuudet luoda sellaisia hahmoja ja tarinoita, joiden kanssa itse viihtyy. Näitä pelatessa ei tarvitse yrittää sivuuttaa oletetulle miespelaajalle valmiiksi pureskeltua roolia. Duke Nukemiin tai Gordon Freemaniin saattaa naisena olla vähän vaikea päästä sisään. Ensimmäisen hahmon tunkkaista macho-olemusta tuskin tarvitsee perustella, mutta jälkimmäinen Half-Lifen sankarikin pelastaa maailmaa vaivaannuttavan läheisissä tunnelmissa duunikaverinsa tyttären kanssa. The Simsissä tai World of Warcraftissa voi sen sijaan itse luoda omaehtoiset merkitykset ja samaistumisen lähtökohdat suhteessa peliin, ilman että tarvitsee luovia pelintekijän linjanvetojen läpi vastavirtaan.

Tietenkään samaistuminen ei ole videopeleissä kaikki kaikessa. Onhan toinen puoli roolipelien viehätystä sitä, että hahmosta voi tehdä myös aivan erilaisen kuin itse on. Eivätkä toistuvasti miespuoliset hahmot aina edes häiritse. Pelaan itse intohimoisesti Team Fortress 2:ta,

jossa kaikki pelattavat hahmot ovat miehiä.

Videopeleissä on paljon sisältöjä, jotka toistuvat uudestaan ja uudestaan, eikä niitä kyseenalaisteta tarpeeksi. Arvosteluissa ja kriittisessä keskustelussa valitellaan kaiken aikaa, kuinka pelit toistavat itseään, eikä tuoretta innovaatiota tapahdu. Valtavirtapelejä markkinoidaan jatkuvasti samoilla tutuilla ja turvallisilla elementeillä, joista peliarvostelijat purnaavat kyllästymiseen asti.

Alkuvuodesta 2012, jo ennen Anita Sarkeesianin myllytystä, muutama internet-yhteisö keksi ottaa silmätikukseen Jennifer Heplerin, erään BioWaren pelisuunnittelijan. Hepler oli 2006 antamassaan haastattelussa kertonut varsin vahvoja mielipiteitä siitä, mitä peleissä pitäisi olla, muttei nykyään ole. Hän kritisoi muun muassa pelien ainaista monotonista taistelemista, kökköä juonenkuljetusta ja huonoa markkinointia. 4chan etunenässä haukkui Kepleriä ”BioWaren syöväksi”.

Keplerillä on ajattelemisen arvoisia pointteja. Hän esimerkiksi ehdotti peleihin laajempaa pikakelausominaisuutta. Dialogikohtauksissa ja välivideoissahan on jo nyt mahdollisuus hypätä yli jorinoita ja tarinankerrontaa. Kepler ehdotti, että samaa voitaisiin soveltaa myös taisteluihin, joita on perinteisesti pidetty tiiviimmin monenlaisten pelien pääasiallisena sisältönä.[6]

Entäs jos kyllästyy taistelemiseen ja haluaisi vain päästä eteenpäin? Minullakin on jo yli puoli vuotta ollut kesken Deus Ex: Human Revolution, koska en vain pääse eteenpäin eräästä pomotaistelusta. Miksei pelissä voisi olla tässä kohdassa skip-nappia? Tappionsa voisi rehdisti myöntää, mutta voisi silti sopia pelin kanssa ”leikitään, että tää kuitenki voitti ton”, ja pääsisi jatkamaan tarinassa eteenpäin.

Ajat muuttuvat

Ei tietenkään ole kenenkään syytä, että tietokonekulttuuri on kauan ollut leimallisesti miesten kulttuuria. Silti on mielenkiintoista miettiä, miksi niin on, ja tarvitsisiko sille tehdä jotain. On menetys sekä alakulttuurille itselleen, jos siitä puuttuvat naiset, ja myös naisille, jos niin monilta jää näin hieno alakulttuuri kokematta.

Muutosta on epäilemättä odotettavissa varsin pian, ja luultavasti sen tuulet puhaltavat kahdelta suunnalta. Yhtäältä feministinen media- ja yhteiskuntakritiikki tukee videopelien kehittymistä tasapuolisemmaksi ja sielukkaammaksi taiteenlajiksi ja auttaa tasoittamaan sukupuolikuilua tietokonealan töissä. Toisaalta taas luova nuoriso tuottaa orgaanisesti ja jatkuvasti uutta kulttuuria uudessa maailmassamme, jossa aidosti henkilökohtaiset tietokoneet ja netti ovat koko ajan läsnä.

Amiraali Grace Hopper (1906-1992) keksi "debuggaamisen" ja COBOL-ohjelmointikielen

On vaikea sanoa, miksi tietokoneet ovat viime vuosikymmenet olleet niin valtavalla marginaalilla miehinen harrastus. Kuitenkin on hyödyllistä pohtia asiaa, vaikkei yksittäisiä selkeästi osoitettavia syitä olisikaan mahdollista nimetä. Tietokoneet ovat valtavan hieno juttu, ja niiden arkisesta käyttäjäkunnasta naisia on nykyään varmasti ainakin puolet. Viisikymppisellä äidilläänikin on nykyään kaksi tietokonetta, läppäri ja tabletti, vaikka takavuosina on käyty useammankin kerran sanaharkkaa siitä, tarvitaanko sinne omaan huoneeseen muka oikeasti oma tietokone, vaikka perheellä yksi beigenharmaa peltipönttö jo olohuoneessa olikin.

Maailma muuttuu.

Lähteet

[1] Tamminen, Tuomo 2006: Poikien välisestä ystävyydestä. Aviisi 02/2006. Tampereen yliopiston ylioppilaskunta. http://www.aviisi.fi/artikkeli/?num=02/2006&id=1733c10

[2] Light, Jennifer S. 1999: When Computers Were Women. Technology and Culture Volume 40, Number 3, July 1999 pp. 455-483. The Johns Hopkins University Press. https://muse.jhu.edu/login?auth=0&type=summary&url=/journals/technology_and_culture/v040/40.3light.html

[3] Tilasto matemaattis-luonnontieteellisen tiedekunnan päävalinnasta 2012. Helsingin yliopisto. http://www.helsinki.fi/ml/valinnat/tilastot/valintatilasto2012.pdf

[4] Feminist Frequency. Conversations with Pop Culture. http://www.feministfrequency.com

[5] Essential Facts About the Computer and Video Game Industry. Sales, Demographic and Usage Data 2012. Entertainment Software Association. http://www.theesa.com/facts/pdfs/ESA_EF_2012.pdf

[6] Polo, Susanna 2012: Inclusion: What Jennifer Hepler's Story Is All About. The Mary Sue 20.2.2012. http://www.themarysue.com/inclusion-what-jennifer-heplers-story-is-all-about/

Needs more ATARI!

Chip-musiikilla on perinteisesti viitattu 1980-luvun tietokoneiden ja pelikonsoleiden äänipiireillä tehtyyn musiikkiin sekä demoskenen piirissä syntyneeseen tyyliin, joka jäljittelee sen soundia. Nykyään "chiptune" tarkoittaa myös 2000-luvun alussa ilmaantunutta suuntausta, joka yhdistelee vanhojen laitteiden ääniä moderniin konemusiikkiin. Haastattelussa modernin chiptune-musiikin kärkinimiin lukeutuva ranskalainen Ultrasyd.

Teksti: Jari Sihvola
Kuvat: Chiptography

S: Mistä sait yllykkeen chiptune-musiikin tekoon?

U: Lapsena minulla oli Amstrad CPC, Game Boy, Sega Game Gear ja Atari Lynx, joista mieltymykseni chip-äänimaailmaan on peräisin. 90-luvulla minulla ei ollut edes nettiyhteyttä, mutta näpertelin Fast Tracker -musiikkiohjelman parissa. 2000-luvulla latasin netistä piraattipelejä koulun atk-luokassa ja löysin cracktrot. Pian eksyin demoskenesivustoille, joiden myötä tartuin taas Fast Trackeriin, ja retropelihararstuksen myötä aloin musisoida myös vanhoilla laitteilla. 8bit collectiven kautta päädyin lopulta chiptuneskeneen.

S: Olet osallistunut demoskenen musiikkikilpailuihin. Onko tavallista, että chipmusic-skenen edustajilla on yhteyksiä demoskeneen?

U: Skenejen välillä on yhteyksiä sekä musiikin, koneiden että yleisemmin kulttuurin tasolla. Toki on myös paljon chiptune-artisteja, jotka eivät ole kiinnostuneita demoskenestä. Löysin itse ensin demoskenen ja vasta sitten uuden chipmusiikin. En ole ainoa lajissani, ja uskon, että yhteydet ovat lisääntymässä. Chiptune-skenestä voisi saada uusia ja tuoreita vaikutteita demopuolellekin.

S: Mistä laitteista nykyajan chiptune-muusikot repivät soundinsa? Game Boy näyttäisi olevan suosituin vehje.

U: Game Boy on tosiaan hyvin suosittu — jopa niin suosittu, että monille chiptune-musiikin historiaa ja demoskeneä tuntemattomille chiptune on yhtä kuin Game Boy. Se on kaikkien tuntema kyvykäs laite, josta saa myös ilman samplejen käyttöä irti monenlaisia ääniä. Game Boyn voi myös kantaa helposti mukanaan kaikkialle. Tietokoneet ovat vähemmän käytännöllisiä ja niitä hyödynnetään vähemmän. Itse käytän eniten Commodore 64:ää, Amstrad CPC:tä, Atari ST:tä ja Game Boyta, joista lavalla pitäydyn kahdessa jälkimmäisessä.

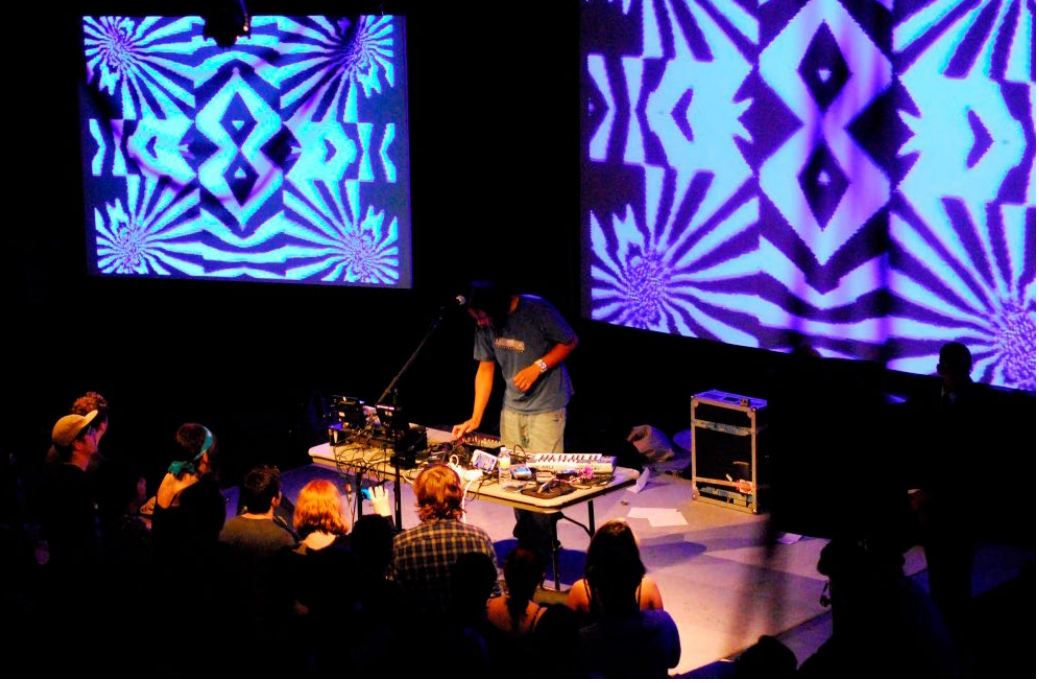
S: Chiptune-musiikkiin kuuluvat myös live-keikat. Mitä artisti tekee esiintyessään livenä?

U: Riippuu artisteista ja käytetyistä ohjelmista. Jotkut eivät miksaile kappaleiden soidessa vaan painavat vain starttia niiden välissä, mutta tuijottavat näyttöä koko setin ajan ikään kuin tekisivät jotain. Nanoloop-ohjelmaa Game Boyssa käyttävät artistit tekevät enemmän, sillä he joutuvat valitsemaan kappaleidensa patternit reaaliajassa. Chiptune-muusikko voi toimia kenen tahansa DJ:n lailla ja lisätä efektejä musiikin päälle Kaoss Padilla tai vastaavalla laitteella. Lisäksi jotkut käyttävät kannettaviaan livenä, soittaen musiikkia emulaattoreista. Sitten on niitä, jotka vain antavat kappaleiden soida ja menevät yleisön eteen hyppimään ja tanssimaan. Pidän tällaista lähestymistapaa hyvin rehellisenä!

U: Itse synkronoin laitteet käsin enkä enää yhdistä Game Boyta ja Atari ST:tä midin kautta. Tarkistan, että kaikki on synkassa, lisään efektejä ja säädän äänen taajuksia. New Yorkissa minulla oli midikoskettimisto, jolla ohjasin yhtä Game Boyta soittaakseni livenä. Luovuin tästä, koska pidän enemmän kevyestä kokoonpanosta enkä ole kovin hyvä soittaja. Voisin myös ohjata Commodore 64:ää tai Game Boyta syntikalla tai kitaralla. Silloin tyylini etääntyisi siitä, mitä kutsutaan electroksi.

S: Atari ST:llä ja sen YM2149 -äänipiirillä on keskeinen rooli musiikissasi. Millainen on suhteesi Atariin? Siinä olevaa äänipiiriä käytettiin alkuperäisenä General Instrumentsin versiona ties kuinka monessa 80-luvun laitteessa.

U: Atari ST oli Amigan ja Amstradin ta-



voin suosittu kotimaassani Ranskassa, mutta ennen vuotta 2009 en ollut koskenutkaan siihen. Ostin tuolloin yhden koneen kokoelmiini ja huomasin sen ääniominaisuuksien olevan samaa ääniipiiriä käyttävää Amstradia laajemmat. YM2149-piirin puhtaaseen soundiin olen mieltynyt enemmän kuin Commodore 64:n kohisevampaan SIDiin. Amigaan verrattuna Atari ST:ssä taas on käytännön etuna tiedonsiirron kätevyys PC:n ja ST:n välillä. Omaan makuuni Atarissa on juuri sopivasti retrotyyliä ja modernimpia ominaisuuksia. Se on vähemmän suosittuna laitteena myös ehkä omaperäisempi valinta.

S: Osa kappaleistasi on tehty 80-luvun äänipiirejä käyttäen, toisissa taas käytät moderneja musiikkiohjelmia. Millaisia kokemuksia sinulla on harrastelijoiden tekemistä ohjelmista musiikin tekoon vanhoille äänipiireille?

U: Amatöörimuusikolle amatöörin tekemä ohjelma, näin sen pitääkin olla. Skenepiireissä tracker-ohjelmia tekevät ihmiset vaikuttavat ammattilaisilta, vaikkei heille makseta työstään. Ohjelmiin tulee jatkuvasti päivityksiä, ja jos haluaa niihin uusia ominaisuuksia, riittää että lähettää tekijöille sähköpostia. Joskus voi päästä ohjelmien beta-testaajaksi ja osallistua kehitykseen suoraan. Ohjelmat ovat yleisesti ottaen hyvin toimivia ja muistuttavat toisiaan: jos osaa käyttää Amigan perinteistä Protrackeria, osaa tehdä musiikkia kaikille koneille.

S: Käytetäänkö chiptune-musiikin tekoon yleensä vanhoja koneita vai PC:n emulaattoreita? Onko oma chiptune-musiikkisi emuloitua vai autenttista?

U: Varsinaiset chiptunet pitäisi säveltää ja soittaa vanhoilla koneilla. Termiä "fakebit" käytetään, kun tehdään chiptune-soundia muistuttavaa jälkeä modernein keinoin. Kun teen musiikkia vanhoille äänipiireille, haluan soundin olevan aitoa. Käytän jonkin verran sampleja, mutta Atari ST soittaa nekin. Käytännön syistä säveltäminen tapahtuu usein PC:llä ohjelmilla, jotka on tarkoitettu 80-luvun äänipiireille säveltämiseen. Niissä on vähemmän bugeja ja työn säilyttäminen on kätevämpää. Lopullinen tuotos äänitetään aidosta koneesta, eli ääni on autenttista.

S: Uudemmat harrastelijoiden suosimat musiikkiohjelmat, kuten Renoise ja Reason, ovat tuoneet tracker-musiikin tekoon lisää efektejä. Ovatko uudet ominaisuudet mukava lisä vai välttämättömyys?

U: Efektien käytöstä huolimatta chiptune on yksinkertaista ja kuulostaa aina hyvältä, koska sillä ei ole muutakaan mahdollisuutta — käytössä on vain pari aaltomuotoa! Moderneilla tracker-ohjelmilla soundin saa komeaksi ja moniulotteiseksi, mutta asioista tulee monimutkaisempia. Jos ei pidä silmällä taajuuksia, jos soitinäänet eivät soi hyvin yhteen tai niihin tulee päällekkäisyyksiä, voi livetilanteessa seurauksena olla katastrofi. Modernien keinojen käyttäjillä on myös virtual sound technology -ohjelmat ja yleisessä levityksessä olevat samplet, joten aina ei tiedä mistä musiikin teko pitäisi aloittaa. Perinteisten chiptune-soundien kohdalla homma alkaa aina samasta tutusta piipityksestä. Perinteisille äänipiireille musiikkia tehdessä on helpompi jäädä "amatööriksi" ja keskittyä vain säveltämiseen. Sen sijaan jokaisen, joka haluaa olla hyvä musiikin tekijä PC:n nykyisillä ohjelmilla, pitää oppia miksaamaan ja jopa masteroimaan — ja siinä riittää pähkäiltävää. Itse teen musiikkia myös PC:n nykyohjelmilla, koska sekin tuntuu kiinnostavalta.

S: Vanhoja äänipiirejä osataan hyödyntää paljon monipuolisemmin kuin 80-luvulla. Onko Koji Kondon ja Jochen Hippelin kaltaisten 80-luvun chiptune-sankareiden suoritukset jo ylitetty?

U: Heidän panoksensa on ollut suuri ja ilman heitä emme olisi päässeet nykytilanteeseen. Chiptune-skenessä alkuperäiset musiikkitiedostot ja soitinäänet leviävät helposti, eikä tekijänoikeutta tunneta. Henki on samantyyppinen kuin demoskenessä, missä koneet pyritään viemään äärirajoille grafiikan ja äänten suhteen, ja saadaan aikaan yhä kehittyneempiä tuotoksia. Musiikin suhteen yleisöön uppoaa helpommin materiaali joka noudattelee nykytrendejä, mutta 80-luvun pioneerisäveltäjien työ on lyömätöntä. Teknisellä puolella on menty eteenpäin, mutta vanhojen sävellysten taika säilyy. Nykyään yritetään tehdä asioita yhä hienommin, mutta samalla niistä tulee mielestäni geneerisiä ja vähemmän omaperäisiä.

Ultrasydin chiptune-artistivinkit:
Je Deviens DJ en 3 Jours, Knife City, Bit Shifter, Sylcmyk, Xyce, Chibitech, NNNNNNNNN, Trash80, Henry Homesweet, Men of Mega, Nullsleep, Bryface, Covox, Trey Frey, Jellica, STU, Radlib, Sulumi, Minusbaby, Deadbeatblast, Hally

256 tavun ohjelmia

QR-viivakoodeja viljellään kyllästymiseen asti. Yleensä niihin on tallennettu nettiosoitteita, mutta niihin saa yhtä hyvin mitä tahansa dataa — vaikkapa ajettavia ohjelmia

Ville-Matias Heikkilä

Skrolli-lehti ikuistaa nyt jälkipolville QR-koodeina yhdeksän hyvin pientä ohjelmaa — kaksi peliä ja seitsemän demoa. Vaikka yhteen kuvioon saakin mahtumaan parhaimmillaan liki kolme kilotavua, ovat tässä julkaistut ohjelmat enintään 256-tavuisia. Koska QR-koodeja harvoin käytetään näin, saattaa ohjelmien toimimaan saattaminen olla yllättävän haastavaa. Pioneerihenkeä siis peliin!

Kaikki ohjelmat yhtä lukuunottamatta ovat MS-DOSille. Lue ohjelman QR-koodi ja tallenna sen suodattamaton raakateksti .COM-päätteiseen tiedostoon. Aja tiedosto joko aidolla PC:llä tai emulaattorilla. Näyttävämmät ohjelmat vaativat enemmän tehoa kuin tyypillisestä DOS-koneesta löytyy, ja esimerkiksi DOSBox-emulaattorin turbomoodia (alt-f12) kannattaa tämän vuoksi käyttää.

Yksi demoista on Javascript-koodinpätkä, joka saattaa jopa toimia joissakin laitteissa ilman kummempia kikkailuja. Se ei ole kovin kaksinen, sillä sen tarkoitus on lähinnä osoittaa konsepti toimivaksi. Javascript on myös tarvitsemansa kehyskoodin määrän vuoksi MS-DOSia vihamielisempi alusta äärimmäisen pienille ohjelmille.

4is256 (Řrřola, 2007) — Tetris-klooni. Ohjaus shift-, ctrl- ja alt-näppäimillä.

Boulder Dash in 256 bytes (James David Chapman, 1995) — Pelkistetty versio klassikkopelistä. Etsi uloskäynti ja varo pudottamasta kiviä päällesi!

Bump is Possible (Downtown, 1999) — Pyörivä betonidonitsi.

Dírojed (Řrřola, 2007) — Takaisinkytkentäpsykedeliaa 32 tavussa.

Javascript-testidemo (Skrolli, 2013) — Yksinkertainen canvas-efekti, jonka jätämme lukijoiden jatkokehitettäväksi!

Puls (Řrřola, 2009) — Animoidut koneiston sisukset raymarchattuna.

Searchlight (Wamma, 2007) — Raycast-katakombi varjostavine pylväineen.

Sqwerz3 (Trimaje, 1996) — Pyörimistä erivärisiten neliöiden seassa.

Tube (3SC, 2001) — Vapaasti pyörivä kamera spiraalitunnelissa.

DWARF FORTRESS

– THE DEVOPS EDITION –

VALOKUITUA!
GENERAATTOREITA!

PALVELIMIA!
LÄHIVERKKOJA!
BGP JA OSPF!

Kun palvelinhuone alkaa täyttyä vihaisista minotaureista, samalla kun lohikäärme katkoo kuitulinkkejä kerrosta ylempänä, saattaa yllätetty sankari katua, että sijoitti palvelimensa kotiluolaansa. Taitaa viikonlop-pu mennä taas korjatessa.

Kapsin jäsenet eivät joudu moisia murehti-maan, kiitos lohikäärmeitä pelkäämättömän rohkean ylläpidon. Ne on aika kovia tyyppejä.

kapsi.fi

TULEVIEN SKROLLIN NUMEROIDEN AIHEITA: ROGUELIKET * IRC * SUOMITIETOTEKNIIKAN HISTORIA * ROMUN HYÖDYNTÄMINEN * LINUX MUSIIKINTEOSSA * AIVOT * SÄHKÖTEKNIIKKAA * DRM ERKKI KURENNIEMI * ESOTEERISET OHJELMOINTIKIELET * EEG LIVEVISUALISAATIO * WARETTAMISEN HISTORIAA * PURE DATA TULEVAISUUS * PIKSELIPORNO * TEKOÄLYOHJELMOINTIA * MUD POLITIIKKAA * HIDASTAKAA SITÄ MOOREN LAKIA * RAYMARCHI * JAILBREAKING * PANKOMPUTATIONALISMI * RAYMARCHING * REALTIME RAYTRACING * SOSIAALISUUDEN ALKEET * KIRJOJA * DYYKKAUSOPAS * PROCESSING * DARK ARTS * OCULUS RIFT *

TERVETULOA PERUSTAMISKOKOUKSEEN
HELSINKIIN 3.8.2013:

SKROLLI RY

PERUSTETAAN HOITAMAAN SKROLLI-LEHDEN ASIOITA!

KAIKKI KIINNOSTUNEET OVAT TERVETULLEITA
OSALLISTUMAAN JA ALOITTAMAAN TOIMINTAA!

TARKEMPAA TIETOA TULOSSA: SKROLLI.FI/RY

MINECRAFT VS BOULDER DASH * JULKAISISINKO INDIE-PELIN * ARDUINO * ATARI 2600 * FREECIV LONGTURN * PDP-1 * K KÄYTTÖLIITTYMÄSUUNNITTELU * KRYPTAUSMENETELMÄT * MUD BASIC * NÄIN SELVIÄT TULEVAISUUDEN VALVONTADYSTOPIASSA UNICODESTA UNOHDETUT * TAPAHTUMARAPORTTEJA * ROOTTAUS * WEBBIPALVELUN PYSTYTTÄMINEN * IRKKAAVAT POLIITIKOT * JONNET EI MUISTA 90-LUVUN MEEMEJÄ * PILVI ON HUONOA DJANGO * LYHYTOHJELMALISTAUKSIA * AIKAETSIVÄ * SARJIS EMULAATIO * SOLUAUTOMAATIT * ROM-HAKKAYS * TOOL-ASSIST BBS * TILAA SKROLLI KAVERILLESIKIN: SKROLLI.FI/TILAA